

06 - 08 - 00

A

# UTILITY PATENT APPLICATION TRANSMITTAL (Large Entity)

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Docket No.  
POU9-1995-0064-US2

Total Pages in this Submission

**TO THE ASSISTANT COMMISSIONER FOR PATENTS**Box Patent Application  
Washington, D.C. 20231

Transmitted herewith for filing under 35 U.S.C. 111(a) and 37 C.F.R. 1.53(b) is a new utility patent application for an invention entitled:

**METHOD AND SYSTEM FOR PROCESSING MESSAGES IN A DISTRIBUTED COMPUTING ENVIRONMENT**

and invented by:

Scott A. Fagen, Richard C. Williams

If a CONTINUATION APPLICATION, check appropriate box and supply the requisite information:

☒ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No.: 08/730,527

Which is a:

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No.:

Which is a:

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No.:

Enclosed are:

**Application Elements**

1. ☒ Filing fee as calculated and transmitted as described below
2. ☒ Specification having 56 pages and including the following:
  - a. ☒ Descriptive Title of the Invention
  - b. ☐ Cross References to Related Applications (if applicable)
  - c. ☐ Statement Regarding Federally-sponsored Research/Development (if applicable)
  - d. ☐ Reference to Microfiche Appendix (if applicable)
  - e. ☒ Background of the Invention
  - f. ☒ Brief Summary of the Invention
  - g. ☒ Brief Description of the Drawings (if drawings filed)
  - h. ☒ Detailed Description
  - i. ☒ Claim(s) as Classified Below
  - j. ☒ Abstract of the Disclosure

# UTILITY PATENT APPLICATION TRANSMITTAL (Large Entity)

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Docket No.  
POU9-1995-0064-US2

Total Pages in this Submission

## Application Elements (Continued)

3. ☒ Drawing(s) (when necessary as prescribed by 35 USC 113)
- a. ☒ Formal                      Number of Sheets 21
- b. ☐ Informal                      Number of Sheets \_\_\_\_\_
4. ☒ Oath or Declaration
- a. ☐ Newly executed (original or copy)                      ☐ Unexecuted
- b. ☒ Copy from a prior application (37 CFR 1.63(d)) (for continuation/divisional application only)
- c. ☒ With Power of Attorney                      ☐ Without Power of Attorney
- d. ☐ DELETION OF INVENTOR(S)  
Signed statement attached deleting inventor(s) named in the prior application,  
see 37 C.F.R. 1.63(d)(2) and 1.33(b).
5. ☒ Incorporation By Reference (usable if Box 4b is checked)  
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied  
under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby  
incorporated by reference therein.
6. ☐ Computer Program in Microfiche (Appendix)
7. ☐ Nucleotide and/or Amino Acid Sequence Submission (if applicable, all must be included)
- a. ☐ Paper Copy
- b. ☐ Computer Readable Copy (identical to computer copy)
- c. ☐ Statement Verifying Identical Paper and Computer Readable Copy

## Accompanying Application Parts

8. ☒ Assignment Papers (cover sheet & document(s))
9. ☐ 37 CFR 3.73(B) Statement (when there is an assignee)
10. ☐ English Translation Document (if applicable)
11. ☒ Information Disclosure Statement/PTO-1449                      ☒ Copies of IDS Citations
12. ☒ Preliminary Amendment
13. ☒ Acknowledgment postcard
14. ☒ Certificate of Mailing

☐ First Class                      ☒ Express Mail (Specify Label No.): EL643175677US

# UTILITY PATENT APPLICATION TRANSMITTAL (Large Entity)

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Docket No.  
POU9-1995-0064-US2

Total Pages in this Submission

## Accompanying Application Parts (Continued)

15. ☐ Certified Copy of Priority Document(s) (if foreign priority is claimed)
16. ☐ Additional Enclosures (please identify below):

## Fee Calculation and Transmittal

### CLAIMS AS FILED

For	#Filed	#Allowed	#Extra	Rate	Fee
Total Claims	12	- 20 =	0	x \$18.00	\$0.00
Indep. Claims	2	- 3 =	0	x \$78.00	\$0.00
Multiple Dependent Claims (check if applicable) <input type="checkbox"/>					\$0.00
BASIC FEE					\$690.00
OTHER FEE (specify purpose)					\$0.00
TOTAL FILING FEE					\$690.00

- ☐ A check in the amount of \_\_\_\_\_ to cover the filing fee is enclosed.
- ☒ The Commissioner is hereby authorized to charge and credit Deposit Account No. 09-0463 (IBM) as described below. A duplicate copy of this sheet is enclosed.
- ☒ Charge the amount of \$690.00 as filing fee.
- ☒ Credit any overpayment.
- ☒ Charge any additional filing fees required under 37 C.F.R. 1.16 and 1.17.
- ☐ Charge the issue fee set in 37 C.F.R. 1.18 at the mailing of the Notice of Allowance, pursuant to 37 C.F.R. 1.311(b).

Blanche E. Schiller  
Signature

Blanche E. Schiller, Esq.  
Reg. No. 35,670  
HESLIN & ROTHENBERG, P.C.  
5 Columbia Circle  
Albany, NY 12203  
Telephone (518) 452-5600  
Facsimile (518) 452-5579

Dated: June 7, 2000

CC:

**CERTIFICATE OF MAILING BY "EXPRESS MAIL"**

In Re Application of: Fagen et al.

Title: METHOD AND SYSTEM FOR PROCESSING MESSAGES IN A  
DISTRIBUTED COMPUTING ENVIRONMENT

Attorney Docket No.: POU9-1995-0064-US2 (0560.049A)

"EXPRESS MAIL" MAILING LABEL NO. EL643175677US

Date of Deposit June 7, 2000

I hereby certify that this paper is being deposited  
with the U.S. Postal Service "Express Mail Post Office  
to Addressee" service under 37 CFR 1.10 on the date  
indicated above and addressed to:

BOX PATENT APPLICATION  
ASSISTANT COMMISSIONER FOR PATENTS  
WASHINGTON, D.C. 20231

CHRISTINE M. ADKINS

(Typed or printed name of person mailing paper or fee)

*Christine M. Adkins*

(Signature of person mailing paper or fee)

Enclosures:

- \* Utility Patent Application Transmittal Letter (3 pages)  
(in duplicate)
- \* U.S. Continuation Patent Application which includes:  
Specification (51 pages), 12 Claims (4 pages), Abstract  
(1 page)
- \* Twenty-one (21) sheets of Formal Drawings
- \* Copy of Declaration and Power of Attorney for Prior  
Application No. 08/730,527 (6 pages)
- \* Copy of Recorded Assignment for Prior Application No.  
08/730,527 (3 pages)
- \* Information Disclosure Citation (1 page) and twelve (12)  
references
- \* Preliminary Amendment (2 pages)
- \* Two (2) Acknowledgment Postcards

002090" 99568560

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Fagen et al.  
Serial No.: Filed Herewith Group Art Unit:  
Filed: Filed Herewith Examiner:  
Title: METHOD AND SYSTEM FOR PROCESSING MESSAGES IN A  
DISTRIBUTED COMPUTING ENVIRONMENT

To: Assistant Commissioner for Patents  
Washington, D.C. 20231

Preliminary Amendment

Dear Sir:

Applicants respectfully request that the following  
amendments be entered, prior to examination of the above-  
referenced application.

Amendments

Please amend the above-identified application, as follows:

In the Specification:

Please amend the specification as follows:

On page 1, after the title, please insert --This application  
is a Continuation Application of prior Application Serial No.  
08/730,527, filed October 11, 1996, now pending.--

On page 10, line 16, delete "colelcted", and replace with  
--collected--.

On page 10, line 24, delete "transation", and replace with --transaction--.

On page 17, line 21, after "Guide", insert --and MVS/ESA SYSPLEX Services--.

On page 17, line 21, before "which", insert --each of--.

On page 49, line 6, before "determine", insert --to--.

**Remarks**

Applicants respectfully request entry of the above amendments and consideration of the application, as amended.

Should the Examiner have any questions regarding this application, please call applicants' attorney at the below listed number.

Respectfully submitted,

Blanche E. Schiller  
Blanche E. Schiller  
Reg. No. 35,670

Dated: June 7, 2000

HESLIN & ROTHENBERG, P.C.  
5 Columbia Circle  
Albany, New York 12203  
Telephone: (518) 452-5600  
Facsimile: (518) 452-5579

-1-

METHOD AND SYSTEM FOR PROCESSING MESSAGES  
IN A DISTRIBUTED COMPUTING ENVIRONMENT

TECHNICAL FIELD

5 This invention relates, in general, to distributed computing environments, and in particular, to a message processing facility within a distributed computing environment, which enables parallel processing of messages used in providing a solution for a request.

10 BACKGROUND ART

Message passing is used for the solution of parallel programming problems or requests. In this use, a network of processes use such a message passing system to communicate to solve the problem.

15 In the current technology of messaging and parallel development support, there is one technique for coordinating such parallel processing topologies, called "scripting." This technique is provided by Workflow products.

20 Scripting provides a technique for describing the topology and conditions for completion to solve a particular problem or flow of work. The script is interpreted and each step may initiate a flow of messages to a set of target tasks (or processes) in  
25 the next step of the topology. The script ends with the description of the conditions for completion. In this way, scripting describes one and only one static topology for the solution of a given problem.

0953956-060700

-2-

Though awareness of the topology is not required in each of the tasks, none of the tasks can conditionally change the topology. This means that variations of a given script must be created for  
5 these conditions, and the conditions must be determined prior to execution so that the proper script for this unique problem can be chosen. This can require considerable effort that in many cases would have been best done when the task encounters  
10 the condition during execution.

There are also a number of roll your own techniques, which are described below. In general, these techniques are designed for single stage parallelism and predetermined topologies. These  
15 techniques provide complicated and easily broken methods for determining the completion of parallel tasks. Using current message passing systems, much of the awareness of the network topology is "hard-coded" in the tasks. Changes to the topology require  
20 changes to the underlying tasks. The current technology is:

1. To create a set of slots which accommodates the predetermined number of expected results for the topology. Each task would  
25 place its result in a predetermined slot. This design requires that only one set of results can be collected for one problem. Otherwise, the result of a subsequent problem might end up filling up or  
30 overwriting a slot for the prior problem. In this methodology, the originating tasks would have to coordinate the initiation of

09509506-050700



-3-

the next problem with the completion of the current problem. This technique is called "cycling."

2. To reproduce the entire network of  
5 processes and queues for each request. In  
this methodology, each problem has its own  
tasks, queues and slots. This solution  
limits the number of concurrent problems  
that can be worked on because the system  
10 must replicate the storage available for  
all the problems and queues used to solve  
each instance of the problem.
3. To place all results in one queue. The  
15 initiating process identifies each new  
problem that it starts and passes that  
identifier with the message to all of the  
parallel processes. Each result is stored  
in the next available slot, not a  
predetermined slot. A completion checking  
20 task looks through all of the collected  
results for all of the problems in process  
and counts the number of results for a  
given problem. When a specified number of  
results are found for that problem, it is  
25 considered complete. Some task is then  
notified to take the results off the queue  
and process them. Clearly, this solution  
has efficiency and resource problems. This  
solution has some obvious contention  
30 problems and possible resource problems.  
If a change is made to the number of  
results expected for a given problem, it is

-4-

now impossible to alter the topology dynamically, as an alteration would affect problems of this type that are currently "in flight." The completion checking routine would have to distinguish between results from problems initiated before and after the alteration.

This problem is usually solved by deferring the initiation of all new problems of this type until the last in flight problem completes, then one informing the completion task of the change in the number of expected results for this problem. Then, new problems of the given type can be initiated and results checked. This is called "cycling the network."

Based on the foregoing, a need exists for an improved message processing facility that allows for the widening of parallel execution of messages. Further, a need exists for a facility that can coordinate the results of a network of processes that has been widened in its parallelism without creating subproblems. Additionally, a need exists for a facility that enables a message to know what request it pertains to and where in the processing sequence of that request it belongs.

#### SUMMARY OF THE INVENTION

The shortcomings of the prior art are overcome and additional advantages are provided through the provision of a message processing facility.

-5-

In one embodiment, a method for processing messages in a computer system is provided. A message is sent to a location designated to receive the message. The message includes a request identifier, specifying a request the message corresponds to, and a sequence identifier, indicating where in a processing sequence of the request a message belongs. The message is then retrieved from the location.

In a further embodiment of the invention, a send function is used to send the message to the location. The send function includes a parameter indicating whether the message corresponds to the request or to another request.

In a further embodiment of the invention, the message is retrieved by a process and the process then sends the message to one or more other processes, which are dynamically determined by the process.

In a further aspect of the invention, a system for processing messages in a computer system is provided. The system includes means for sending a message to a location designated to receive the message and means for retrieving the message from the location. The message includes a request identifier, specifying a request the message corresponds to, and a sequence identifier, indicating where in a processing sequence of the request the message belongs.

The method and system of the present invention advantageously saves storage, processing power and

00589566-060700

-6-

simplifies the programmer effort. It provides high level services that allow the creation of parallel networks of processes without requiring process awareness of the topology or the position a given process may have in the topology. Processes or networks need not be cycled or stopped. Because the message carries an identifier of the request it is a part of and its unique location, the nexus for the request it is a result for can be determined automatically. The width of the parallelism involved in a particular request is tracked, and therefore, the appropriate number of results expected can be determined automatically and dynamically. Therefore, despite dynamic changes in topology, completion can be determined without awareness of the processes that create the results.

Other embodiments and aspects of the invention are described in detail herein and are considered a part of the claimed invention.

20

#### BRIEF DESCRIPTION OF THE DRAWINGS

The subject matter which is regarded as the invention is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the invention will be apparent from the following detailed description taken in conjunction with the accompanying drawings in which:

-7-

FIG. 1 depicts one example of a sysplex incorporating and using the principles of the present invention;

5        FIG. 2 depicts one example of a vector table used in accordance with the principles of the present invention;

FIG. 3 depicts one example of an index table used in accordance with the principles of the present invention;

10       FIGS. 4a-4b depict examples of a distribution table used in accordance with the principles of the present invention;

15       FIG. 5 depicts one example of a queue table used in accordance with the principles of the present invention;

FIG. 6 depicts one example of the fields associated with a message sent and received, in accordance with the principles of the present invention;

20       FIG. 7 depicts one example of the fields associated with a nexus table used in accordance with the principles of the present invention;

25       FIGS. 8a-8b depict one embodiment of the logic associated with initializing the message processing mechanism of the present invention;

00589566-060700

-8-

FIGs. 9a-9c depict one embodiment of the logic associated with cleaning-up when PQM is terminated, in accordance with the principles of the present invention;

5           FIG. 10 depicts one embodiment of the logic associated with loading the distribution table of FIGs. 4a-4b, in accordance with the principles of the present invention;

10           FIG. 11 depicts one embodiment of the logic associated with loading the queue table of FIG. 5, in accordance with the principles of the present invention;

15           FIG. 12 depicts one example of a queue, in accordance with the principles of the present invention;

20           FIG. 13 depicts one example of the logic associated with initializing a number of service tasks to be used by users of the message processing facility of the present invention, in accordance with the principles of the present invention;

25           FIG. 14 depicts one embodiment of the logic associated with establishing a connection to a queue for sending to or receiving messages from the queue, in accordance with the principles of the present invention;

FIG. 15 depicts one embodiment of the logic associated with removing connection to a queue

09589566-050700

-9-

established by the logic of FIG. 14, in accordance with the principles of the present invention;

5        FIGs. 16a-16h depict one embodiment of the logic associated with sending a message to a queue opened by the logic of FIG. 14, in accordance with the principles of the present invention;

10        FIG. 17 depicts one example of the logic associated with resolving a list of queue names to a list of system identifiers and address pairs, in accordance with the principles of the present invention; and

15        FIG. 18 depicts one embodiment of the logic associated with receiving messages placed in a queue by the logic of FIGs. 16a-16h, in accordance with the principles of the present invention.

BEST MODE FOR CARRYING OUT THE INVENTION

20        In accordance with the principles of the present invention, a mechanism for processing messages in a computer system is provided. A message is placed on (or sent to) a queue. Messages are taken off a queue by processes that are designed to read these queues,  
25        process the message and send the message to another queue or to multiple queues. When a message is sent to multiple queues or "fanned out", the receiving processes may each receive and separately and simultaneously do work on a part of the results for a

002090-99658560

given problem. Such a multiple send initiates a transaction and the original transaction is called a request. Each message carries, for example, a triplet of a request identifier, a transaction identifier and a parallelism width identifier, called a limb number, as well as the application data. A process receiving one of these messages may do some processing and modify the application data in the message. The process can then send its result on to another queue or declare that the transaction limb is complete.

In a further aspect of the invention, a process on a limb of a transaction may receive a message and process the message using parallel processes. To do this, the process can create a new transaction (also called a subtransaction). The nexus for this subtransaction records the information necessary to collect the results of the subtransaction and properly restore the triplet so that the result of the subtransaction can be treated as the limb that created it and be collected in the nexus of the next outermost transaction. This technique is called nesting transactions.



-11-

09589566-060700

In a further embodiment of the invention, it is also possible for a process to add more limbs to the transaction it is a part of, rather than nest transactions. A transaction containing a process  
5 that does this is called an eccentric transaction. The invention allows the number of limbs to be added to the eccentric transaction to be determined by the process at execution time. For this, the nexus for the eccentric transaction can be altered to handle  
10 more limbs. Each message on the new limbs receives, for example, the appropriate triplet to uniquely identify its place in the transaction. Completion of the transactions waits until all of the results are collected from all of the limbs including the new  
15 ones.

In a further aspect of the invention, the collected results are combined into a new message and a triplet is created from the transaction information in the nexus. If the transaction that the nexus  
20 served was created as a subtransaction by a limb of another transaction (called the next outermost transaction), then a triplet is formed which identifies this new message as being that limb and a part of the next outermost transaction and request.

25

FIG. 1 depicts one example of a computing environment 100 incorporating and using the message processing facility of the present invention. In one example, computing environment 100 includes a  
30 plurality of computing systems 102 and a coupling facility 104, each of which is described in detail below.

-12-

As one example, each computing system 102 is designed following the Enterprise Systems Architecture/390 offered by International Business Machines Corporation, as described in Enterprise Systems Architecture/390 Principles of Operation, IBM Publication No. SA22-7201-02 (December 1994), which is hereby incorporated herein by reference in its entirety. As shown, each computing system executes at least one operating system, such as, for instance, the Multiple Virtual Storage (MVS) operating system offered by International Business Machines Corporation, which is used to control execution of programs and the processing of data, as is well known. Also, in accordance with the principles of the present invention, each system includes a message processing facility used in the receiving, processing and sending of messages, as described in detail below.

Each of the computing systems is coupled to coupling facility 104 via one or more possible connections, such as Cross-System Extended Services (XES) software. Coupling facility 104 is described in detail in MVS/ESA SYSPLEX Services Guide (IBM Publication No. GC28-1495-02) (June 1995) and MVS/ESA SYSPLEX Services and Reference (IBM Publication No. GC28-1496-02) (June 1995), which are hereby incorporated herein by reference in their entirety. In one embodiment, coupling facility 104 is a structured-external storage processor that enables data and processing facilities to be shared by all of the systems connected to the coupling facility.

-13-

00589566-060700

The above-described computing environment incorporating and using the message processing facility of the present invention is only one example. The techniques of the present invention can be used with many types of computing environments. For example, the invention can be used within an environment having one system coupled to the coupling facility or many systems coupled thereto. Further, a computing system can execute one or more operating systems different from MVS. Additionally, an external coupling facility other than the one described herein can be used without departing from the spirit of the invention. Additionally, the message processing facility of the present invention can be used in a single system environment that does not have or need a coupling facility. Thus, the techniques of the present invention can be used with a computing environment having a single computing system with one or more operating systems, or a sysplex environment having a plurality of computing systems, or any other type of systems, without departing from the spirit of the present invention.

In accordance with the principles of the present invention, a number of data structures are used in the message processing facility of the present invention. An example of each of these data structures is described in detail below.

One data structure includes a vector table 200 (FIG. 2), referred to herein as PQMVT, which is used in keeping track of some of the other data structures of the present invention. In one embodiment, there is a vector table on each system wishing to

-14-

participate in the message processing facility of the present invention and each PQMVT 200 includes the following fields:

- 5           (a) An eye catcher field 202 having a value of PQMVT specifying that this is the PQMVT table;
- 10           (b) A PQMDTM address 204 indicating the address of a local distribution table, PQMDTM, having one or more distribution entries, as described in detail below;
- 15           (c) A PQMQTBL address 206 indicating the address of a local queue table, PQMQTBL, having one or more queue entries, as described in detail below;
- 20           (d) A PQMQANDX address 208 indicating the address of an queue index table, PQMQANDX, which includes a list of the queues open for each process of each address space of the computing environment, as described in further detail below;
- (e) A transaction-counter 210 representing a count of the number of transactions started on the system;
- 25           (f) A request-counter 212 representing a count of the number of requests issued on the system; and

09569566-060700

-15-

- (g) A PQM available flag 214 indicating whether the message processing facility of the present invention is available on the system.

5 Another data structure used in accordance with the principles of the present invention is PQMQANDX 300 (FIG. 3). PQMQANDX is an index identifying which queues are open for which processes executing in which address spaces of the computing environment.  
10 This data structure is primarily used during the clean-up phase of the invention and includes, for example, a header 302 and one or more entries 304, each of which is described in detail below.

15 In one embodiment, header 302 includes the following fields:

- (a) An eye catcher field 306 indicating this is PQMQANDX;
- (b) A number of entries field 308 representing how many entries are included in entries 304, described below; and  
20
- (c) An address and length designation 310 for each entry of PQMQANDX.

In addition to the above, each entry 304 of PQMQANDX 300 includes, for example:

- 25 (a) An address space identifier (ASID) 312 specifying an identifier of the user's address space having a connection to one or

00589566-060700

- 16 -

more queues specified in the entry, as described below;

- (b) A task token 314 specifying the process within the user's address space that has a connection to the one or more queues specified in the entry; and
- (c) One or more queue names 316 that are open for the ASID/task token indicated in the entry.

Another data structure used in accordance with the principles of the present invention includes a distribution table, which represents a directory of all of the possible destinations for messages. A destination may be a queue name or a list of queues names. In accordance with the present invention, the distribution table includes one or more distribution list names, each of which includes one or more queue names, as described below. One example of a distribution table, PQMDTM, is described in detail below.

In one embodiment, PQMDTM 400 (FIG. 4a) includes a header 402 and one or more entries 404, each of which is described in detail below.

As one example, header 402 of PQMDTM 400  
25 includes the following fields:

- (a) An eye catcher 406 identifying this table as PQMDTM;

-17-

- (b) A number of entries field 408 indicating a count of the number of entries in entries 404; and
- (c) An address and length designation 410 for each of the entries.

Each of entries 404 includes, for instance:

- (a) A distribution list name 412 indicating the name of the distribution list; and
- (b) A list of queue names 414 representing all of the queues represented by the associated distribution list name.

The above-described PQMDTM is located in local storage on each of the systems executing the message processing facility of the present invention.

However, in addition to the local copies, PQMDTM is stored in the coupling facility in a list structure 416, as shown in FIG. 4b. By definition, PQMDTM list structure 416 is accessible by each of the systems coupled to the coupling facility. List structures are described in detail in MVS/ESA SYSPLEX Services Guide and Reference, which is hereby incorporated herein by reference in its entirety.

In addition to the above-described data structures, another data structure, referred to as PQMQTBL, is used, in accordance with the principles of the present invention to identify the queues designated for the local system. Each of the queues is capable of receiving messages, which can later be

-18-

retrieved for processing. Each system participating in the message processing facility of the present invention has its own PQMQTBL. In one embodiment, PQMQTBL 500 includes a header 502 and one or more  
5 entries 504, each of which is described in detail below.

In one example, header 502 includes the following fields:

- 10 (a) An eye catcher 506 designating this table as PQMQTBL;
- (b) A number of entries field 508 indicating how many entries are included in entries 504; and
- 15 (c) An address and length designation 510 for each entry of the queue table, PQMQTBL.

Additionally, each entry of PQMQTBL includes, for example:

- (a) A queue name 512 identifying the queue;
- 20 (b) An address 514 specifying where the queue identified by queue name 512 is located in storage; and
- (c) An Event Control Block (ECB) chain 516 indicating pointers to the ECBs waiting for this queue.

00589566-060700



Messages are stored in and retrieved from queues, as described in detail herein. Each message 600 (FIG. 6) includes a number of fields, each of which is described in detail below:

- 5 (a) An eye catcher 602 indicating that this is  
a message;
- (b) An offset to message-text 604 indicating  
where in storage the text of the message is  
located;
- 10 (c) A tran-id 606, which is a unique  
identifier, identifying a transaction.  
Tran-id 606 gets assigned whenever a  
transaction is started;
- (d) A req-id 608 identifying the request for  
15 which the transaction is being processed.  
A request may, in accordance with the  
principles of the present invention, have  
one or more transactions associated  
therewith;
- 20 (e) A limb # 610 identifying the number of  
limbs within the nexus that the message is  
associated with, as described in detail  
below;
- (f) A replyto sysid/queue 614 indicating the  
25 name of the queue to receive the results of  
processing this limb;

-20-

- (g) A nexus address 616 specifying an address of a nexus data structure (described below) associated with the transaction id specified in the message; and
- 5 (h) A message-text 618 indicating the text of the message to be processed, as described in further detail below.

09585566-060700

All of the message fields, except for the message text, are collectively referred to herein as message header 620. In accordance with the principles of the present invention, the message header is not completed by the user but filled in during the processing of the present invention. The message header includes the context of the message processing, and thus, the values of the fields of the message header are carried forward from a received message to messages that the receiving process sends in the present invention.

10

15

.. In addition to the data structures described above, another data structure referred to as a nexus data structure is used, in accordance with the principles of the present invention, to manage nested transactions (a fan-out) of one request. In one embodiment, a nexus data structure 700 (FIG. 7) is created each time a new transaction is started and each nexus data structure includes, for instance, a header 702 and one or more nexus entries 703, each of which is described in detail below.

20

25

In one example, header 702 of nexus 700 includes the following fields:

30

-21-

- 09585555-060700
- (a) An eye catcher field 704 indicating that this is a nexus data structure;
  - (b) An offset to pointer (ptr) list 706 indicating an offset to the nexus entries pointer list field 720 (described below) located within the nexus header;
  - (c) A number of limbs field 708 specifying the number of limbs included in the transaction specified by tran-id 606 (FIG. 6), as described in further detail below;
  - (d) An entries needed to complete field 710 indicating the number of entries needed to complete the processing of the transaction represented by tran-id 606;
  - (e) A next-outer-tran-id field 712 specifying the transaction being processed when the transaction represented by tran-id was created. This shows a nesting of transactions and enables the process to get back to the previous transaction;
  - (f) A next outer queue name 714 indicating the queue that is associated with the transaction identified in the next-outer-tran-id field;
  - (g) A next-outer-nexus address 715 indicating the nexus address for the nexus of the next outermost transaction;

-22-

- (h) A next-outer-limb number 716 specifying the limb of the transaction in process when this new transaction created this nexus;
- 5 (i) A queue name for collected results field 718 indicating the target queue for the results of the transaction when the transaction is complete; and
- 10 (j) A nexus entries pointer list 720 specifying the address of the entries list of the nexus data structure.

Additionally, each of nexus entries 703 includes, in one example, the following fields:

- (a) A length field 722 indicating the length of the entry; and
- 15 (b) An entry 724 including the message header and message text that will hold the final message (the result) from this limb of the transaction.

20 In accordance with the principles of the present invention, in one embodiment, prior to a process being able to deliver or receive any messages, the message processing facility of the present invention is enabled. In particular, in one example, a started task, referred to as PQM, is started on each of the

25 computing systems wishing to participate in the message processing facility. For instance, an operator of the system, or a hot list, is used to start PQM. The hot list can automatically start PQM

00589966-060700

-23-

at prespecified times. PQM enables messages to be forwarded and received, in accordance with the present invention, by initializing certain fields used by the invention and setting up the environment, as described below. In one example, PQM includes job control language (JCL) used in executing an initialization routine, referred to as PQMINIT. The parameters for PQMINIT include, for instance, VT indicating in megabytes (MB) the size of the vector table and VERS indicating a two digit version suffix for parmlib members, used in the manner described below.

One embodiment of the logic associated with PQMINIT is described below with reference to FIGs. 8a-8b. Initially, common storage is allocated on the system executing PQMINIT for the vector table, PQMVT (200), STEP 800. The amount of storage allocated is based on the VT parameter passed with PQMINIT. Additionally, the fields of the vector table are initialized to their default values (e.g. zero), and a name token, PQMMVT, is created and set to the address of PQMVT 200, STEP 802. The name token, PQMMVT, can be used by any task in the system to locate the vector table. Thereafter, an exclusive enqueue is taken on the initialization routine by enqueueing on a resource name, referred to as SYSPQM.INITIALIZE, STEP 806. The exclusive enqueue enables the process performing the initialize routine to complete it without concern of another process also entering the routine and possibly corrupting tables used during that routine, such as PQMDTM.

-24-

Subsequent to successfully setting the exclusive enqueue, a Call IXCQUERY is performed to receive requested status from the coupling facility, STEP 808. One example of Call IXCQUERY is described in detail in "MVS/ESA: Programming: Sysplex Services Reference MVS/ESA System Product: JES2 Version 5 JES3 Version 5," IBM Publication Number GC28-1496-02, June 1995, which is hereby incorporated herein by reference in its entirety. Call IXCQUERY is used, for instance, to determine if the system running the initialization process is coupled to the coupling facility (i.e., part of a sysplex, where the sysplex has one or more computing systems coupled to the coupling facility), INQUIRY 810. If the system is in the sysplex, then a further determination is made as to whether a group referred to as PQMGRP is defined in the sysplex, INQUIRY 812. PQMGRP is a group of all the computing systems of the sysplex that wish to be a part of managing the messaging processing of the present invention (i.e., the PQM managers). Should the group not be defined indicating that this is the first system to execute PQMINIT, then the group is defined, STEP 814. In one example, the group is created as a list managed by the Cross-System Coupling Facility (XCF). One example of creating groups in a sysplex is described in detail in MVS/ESA SYSPLEX Services Guide and Reference, which are incorporated herein by reference in their entirety, as noted above. Each element of such a group is called a member.

Once the group, PQMGRP, is defined or if it had previously been defined, INQUIRY 812, then the system joins the group by executing an IXCJOIN macro, STEP

-25-

816. The IXCJOIN macro places a XCF group (for example, the system) in the active state, associating it with a XCF group, such as PQMGRP. In the active state, the member can use the monitoring and signalling services of XCF. One example of the IXCJOIN macro is described in detail in "MVS/ESA:Programming: Sysplex Services Reference MVS/ESA System Product: JES2 Version 5 JES3 Version 5," IBM Publication Number GC28-1496-02, June 1995, which is hereby incorporated herein by reference in its entirety.

In addition to the above, the joining of the group allows a message exit to be specified, which is used by XCF to send messages to a remote system in accordance with the principles of the present invention, as described in detail herein. In one example, the message exit is referred to herein as PQMXXM, which is described in detail below.

Thereafter, an inquiry is made to determine if this is the first PQM started in the sysplex, INQUIRY 818. In other words, is there just one entry in the PQMGRP. If this is the first entry, then a distribution table manager, referred to as PQMDTM, which is used to establish the local distribution table (DTM) on the system, is called with parameters INITSYSP, and VERS, STEP 820. INITSYSP indicates that the local distribution table is loaded from the parmlib and is copied to the coupling facility; and VERS indicates the two digit version suffix for the parmlib. PQMDTM is described in further detail below.

-26-

If, on the other hand, this system is not the first queue manager in the sysplex, then PQMDTM is called with the VERS parameter and an UPDATE parameter indicating the local distribution table is loaded from the coupling facility, STEP 822. Additionally, if the system is not in the sysplex, INQUIRY 810, then PQMDTM is called passing the following parameters: an INIT parameter, indicating that the local distribution table is loaded from the parmlib and kept locally (no copy at the coupling facility), and the VERS parameter, STEP 824.

After the local distribution table is loaded in accordance with the parameters passed during the call, the exclusive enqueue on SYSPQM.INITIALIZE is freed, STEP 825 (FIG. 8b). Additionally, an indication is provided that when PQM exits for any reason, several clean-up routines are to be called in the following order: EOMMS, EOMPQM and EOMINIT. Each of the clean-up routines is described below. In addition to indicating the appropriate end of memory (EOM) exits to be used when PQM terminates, PQMQMGR, which is used to manage the queues listed in the queue table, is attached. Thereafter, the initialization procedure is complete.

In one embodiment, EOMMS is used to indicate on each system of the computing environment that PQM is no longer active, STEP 900 (FIG. 9a). This is indicated in PQMVT. Additionally, storage for PQM is freed, except for PQMVT.

In addition to EOMMS, the EOMPQM routine is called on termination to post all the ECBs located on

00509566-060700



-27-

the system having the terminated PQM. One example of the logic associated with EOMPQM is depicted in FIG. 9b. Initially, PQMQANDX is looped through to find all of the queue names for the ASID and task token of the terminated PQM, STEP 904. With this information, the entries in PQMQTBL are located and all of the ECBs for the relevant entries are posted, STEP 906. This enables pqmrecv(), discussed below, to detect that the PQM address space has ended and give a termination return code to the caller.

The cleaning up process also includes EOMINIT used to set the PQMMVT name token to zero, to remove the terminated PQM from the PQMGRP, and release storage from PQMVT, STEP 908 (FIG. 9c).

As previously mentioned, during the initialization process, PQMDTM is called in order to load or update the local distribution table. The parameters for PQMDTM include, the function, such as INIT, INITSYSP, or UPDATE; and VERS, the two-digit version suffix passed from PQMINIT. One example of the logic associated with PQMDTM is described in detail with reference to FIG. 10.

Referring to FIG. 10, initially a determination is made as to whether the function passed as a parameter is equal to INITSYSP, INIT or UPDATE, INQUIRY 1000. If it is equal to INITSYSP or INIT, then an exclusive enqueue is taken on a resource, referred to as SYSPQM.DISTRIBUTION\_TABLE, such that the local distribution table and/or the distribution table in the coupling facility can be updated without corrupting either of the tables, STEP 1002.

-28-

Thereafter, storage is allocated for the local distribution table and the address of the local distribution table is placed in PQMVT 200 at PQMDTM address 204, STEP 1004.

5           Subsequent to allocating storage for the local distribution table, distribution table entries previously stored in the parmlib having the version number indicated by VERS are read and stored in the local distribution table, STEP 1006. Thereafter, if  
10   the function passed in during the call is INITSYSP, INQUIRY 1008, then PQMDTM connects to a XES structure (i.e., the DTM list). Then, the PQMEEXIT is specified as the event handler, STEP 1009. Thereafter, the entries are read from the local  
15   distribution table and stored in the distribution table located in the coupling facility, which in one example is a list structure, STEP 1010. This update triggers the event exit PQMEEXIT on other systems in the SYSPLEX. In particular, PQMDTM(UPDATE) is called  
20   on those systems. After the storing is complete or if the parameter function is equal to INIT, INQUIRY 1008, then the exclusive enqueue on SYSPQM.DISTRIBUTION\_TABLE is freed, STEP 1012 and PQMDTM is complete.

25           Returning to INQUIRY 1000, if the requested function is UPDATE indicating that the distribution table at the coupling facility has been changed, then the local distribution table is updated from the table in the coupling facility. Therefore, flow  
30   passes to STEP 1020 in order to effect this change. In particular, in one example, a shared enqueue is taken on SYSPQM.DISTRIBUTION\_TABLE, STEP 1020, and

00589566-060700

-29-

the entries are read from the distribution table in the coupling facility into the local distribution table, STEP 1022. Thereafter, the shared enqueue is freed, STEP 1024, and PQMDTM is complete.

5           After the distribution tables have been loaded and updated, as requested, then as described above with reference to FIG. 8a, the queue manager, PQMQMGR is started by PQMINIT. In one example, PQMQMGR is initiated via, for example, an MVS ATTACH macro.

10   (One example of the ATTACH macro is described in detail in MVS/ESA Programming: Authorized Assembler Services Reference (ALESERV-DYNALLOC), IBM Publication No. GC28-1475-02, September 1995, which is hereby incorporated herein by reference in its

15   entirety. The ATTACH macro passes the version number, VERS, from PQMINIT, which is used during the processing of PQMQMGR.

One example of the logic associated with PQMQMGR is described in detail with reference to FIG. 11.

20   Initially, in one embodiment, storage is allocated for the queue table, PQMQTBL, STEP 1100. Thereafter, all of the queue entries are read from the PQMDTMxx (where xx is equal to VERS) parmlib member and placed in the queue table, STEP 1102. Additionally, the

25   address of the queue table is placed in PQMVT 200 at PQMQTBL address 206 to provide addressability for the queue table, STEP 1104.

Thereafter, a queue is created for each queue entry in PQMQTBL, STEP 1106. In one example, a queue

30   1200 (FIG. 12) includes a queue header 1202 and one or more queue entries 1204, each of which is

05589556-060700

described in detail below. Queue header 1202 includes, for instance, the following fields:

- 10

15

20

25

-31-

Subsequent to creating the queues, a routine referred to as PQMSMGR is started in order to load the code that processes messages using the technique of the present invention, STEP 1108. One example of the logic associated with PQMSMGR is described below with reference to FIG. 13. Initially, common storage is allocated for the PC authorization table used for accessing certain C library routines, described in detail below, STEP 1300 (FIG. 13). Additionally, code is loaded for PQMOPEN used in establishing a connection to specified queues; PQMCLOSE used in disconnecting the established connection; PQMSEND used in sending a message to one or more queues; and PQMRECV used in retrieving a message from the queue, as described in detail below, STEP 1302.

After the code is loaded, the addresses of the loaded code are stored in the PC authorization table, STEP 1304, and PQM is indicated as active in PQMVT, STEP 1306.

Described above in detail is one embodiment for initializing the data structures and managers used in the message processing facility of the present invention. Once the initialization procedures are complete, users can open and close a queue, as well as place messages in and retrieve messages from a queue. In one example, the tasks that enable the above functions are referred to as: PQMOPEN, PQMCLOSE, PQMSEND, and PQMRECV, respectively, each of which is described in detail below.

PQMOPEN is initiated by, for instance, a Program Call executed within a user process, pqmopen(). (In

-32-

one example, pqmopen() is a C library routine.)  
PQMOPEN is used to create a connection between the  
user process and a queue for the purpose of reading  
from and writing to the queue. The process pqmopen()  
5 includes a Q\_NAME parameter indicating the queue that  
the user wishes to connect to. One example of the  
logic associated with PQMOPEN is described in detail  
below with reference to FIG. 14.

Initially, the Q\_NAME received as a parameter  
10 from pqmopen() is searched in PQMQTBL 500, STEP 1400.  
If the requested queue name is not located in the  
queue table, INQUIRY 1402, then a return code is set  
to NOT\_FOUND, STEP 1404, and a Program Return with  
the return code is performed from the PQM address  
15 space back to the user's address space, STEP 1406.  
If, on the other hand, the queue name is found in the  
queue table, INQUIRY 1402, then an exclusive enqueue  
is taken on the resource name SYSPQM.QTBL\_<Q\_NAME>,  
STEP 1408. This enqueue allows an ECB to be created  
20 and added to the ECB chain for this queue name  
without other processes trying to open the same queue  
at the same time.

Subsequent to obtaining the exclusive enqueue,  
an ECB is created for the queue name and the address  
25 space identifier (ASID) of the address executing this  
user process, STEP 1410. The ECB is created in  
extended common storage area (ECSA) and the wait bit  
of the ECB is OFF. Additionally, the address of this  
ECB is added to the end of the ECB chain for this  
30 queue name in PQMTBL 500, STEP 1412.

00589566-060700

-33-

Next, the address space identifier and task token for the process issuing the PQMOPEN are obtained, STEP 1414. For example, in order to obtain the ASID, one queries the PSA and locates the ASCB pointer and follows this to the ASID field in the ASCB. This operation is described in the MVS/ESA Data Areas, Vol. I, LY28-1857. Further, to find the "task token," a TCBTOKEN macro is used, which is described in MVS/ESA Programming: Authorized Assembler Services Reference (SETFRR-WTOR), GC28-1478-01, June 1995, which is hereby incorporated herein by reference in its entirety. Then, the address space identifier, task token and queue name are placed in PQMQANDX, STEP 1416. Thereafter, the exclusive enqueue on SYSPQM.QTBL\_<Q\_NAME> is released, STEP 1418, and the return code is set to SUCCESS, STEP 1420. Subsequently, a Program Return back to the user's address space with the return code is performed, STEP 1406.

PQMCLOSE is used to remove interest in a particular queue for the calling task. When the user wishes to remove interest in a particular queue, a C library routine, pqmccls(), is used to issue a Program Call to PQMCLOSE. In one example, pqmccls() includes a Q\_Name\_List parameter specifying a list of the queues in PQMQTBL to be closed. One example of the logic associated with PQMCLOSE is described in detail with reference to FIG. 15.

Initially, in one embodiment, the appropriate entry 304 (FIG. 3) of PQMQANDX 300 is obtained using the address space identifier and task token of the process issuing the PQMCLOSE, STEP 1500. Next, a

00589566 060700

-34-

determination is made as to whether the Q\_Name\_List parameter is null indicating that no queue name was specified, INQUIRY 1502. If it is null, then a list of queue names (Q\_Name\_List) is created from  
5 selecting the queue names from the PQMQANDX entry obtained in STEP 1500, STEP 1504.

Thereafter or if the Q\_Name\_List parameter is not null, a further check is made to see if the Q\_Name\_List is empty, INQUIRY 1506. Should the  
10 Q\_Name\_List be null, then a successful return code is passed to the user indicating that all the queues for the user's process are closed, STEP 1508.

However, if the Q\_Name\_List is not null, then the head of the Q\_Name\_List is popped, STEP 1510, and  
15 an exclusive enqueue is taken on SYSPQM.QTBL\_<Q\_name>, STEP 1612. Next, the queue name is removed from PQMQANDX, STEP 1514, and the ECBptr for this address space identifier is removed from PQMQTBL for this queue name, STEP 1516.  
20 Thereafter, the storage in ECSA for this ECB is freed, STEP 1518, the enqueue on SYSPQM.QTBL\_<Q\_name> is freed, STEP 1520, and processing continues with INQUIRY 1506.

In addition to opening and closing queues, the  
25 message processing facility of the present invention provides for the sending of messages to open queues. PQMSEND is used to place messages in queues so that processes can retrieve the messages using, for example, PQMRECV, and perform the tasks specified by  
30 the messages. In one example, PQMSEND is Program Called by a user's C library routine, pqmsend().

00589566-060700



-35-

Pqmsend() includes a number of parameters, such as Q\_Name\_List specifying one or more queue names to receive the message, a message\_list specifying a list of messages, an association between messages and queue names and an option parameter that can have a number of different values. For example, the options include REPLYTO(), which indicates a transaction is being initiated for a particular request and a nexus (a.k.a., a collection list) is needed. The argument to the REPLYTO() indicates the queue that should be sent the collected results when a particular transaction is complete. A message with the collected results is sent to the queue specified in the REPLYTO parameter. Another option includes <null> or \* indicating no specific queue name is provided by the user; or REPLY indicating that the message is to be sent to the nexus as one of the results specifying an end of the limb's processing. One embodiment of the logic associated with PQMSEND is described in detail below with reference to FIGs. 16a-16h.

Initially, a determination is made as to whether a null is provided in the Q\_Name\_List, INQUIRY 1600. If null was specified, then no specific queue name was provided by the user, so a target-queue-name is obtained from the message-header of the message to be stored in the queue. In particular, the target-queue-name is retrieved from replyto sysid/queue field 614 of the message-header, STEP 1602. The extracted queue name is then stored in the Q\_Name\_List parameter of the PQMSEND, STEP 1603. Further, in one embodiment, if a null queue name is provided, then it is assumed that the end of a limb

has been reached, therefore, the REPLY option is added to the option list, STEP 1604.

10

15

20

25

30

-37-

header are filled in during the message processing of the present invention, as described herein.

Subsequent to creating the message header or if the header previously existed, the Q\_Name\_List parameter is resolved to a list of system ids and queue names or local queue addresses, STEP 1620. In one example, the Q\_Name\_List includes names of one or more distribution lists in PQMDTM having one or more queue names, or one or more queue names from PQMQTBL or any combination of the two, and thus, this list is broken down (i.e., resolved) into a simple list of queues, including the system (sysid) and address of each queue. A RESLV routine is used for this task and includes as parameters the Q\_Name\_List and a buffer\_ptr indicating an address of storage for the result of the resolve process. One embodiment for performing this resolve is described in detail below with reference to FIG. 17.

In one embodiment, the addresses of PQMDTM and PQMQTBL are extracted from PQMVT 200, and a list to contain the queue sysids and addresses (i.e., LIST) is created and initialized to null, STEP 1700.

Subsequent to initializing req-id 608 and LIST, a check is made to determine if the Q\_Name\_List passed as a parameter from PQMSEND is null, INQUIRY 1702. If the list is null, then the process is ended, since there is nothing to resolve, STEP 1703. However, if the Q\_Name\_List is not null, the head of the Q\_Name\_List is removed and this queue name is searched for in PQMDTM, STEP 1704. If the queue name is found in PQMDTM indicating that the queue name

represents a distribution list, INQUIRY 1706, then the distribution list corresponding to that name is extracted from the local PQMDTM, STEP 1708.

5  
10

15  
20

25  
30

-39-

includes all of the same fields as message header  
620.

Next, a determination is made as to which option  
is specified on the options parameter of the PQMSEND,  
5 INQUIRY 1626 (FIG. 16b). If the REPLYTO() option is  
specified, then a transaction is being initialized  
and a nexus structure is set up for the new  
transaction. A unique transaction identifier is  
assigned, STEP 1628. This transaction identifier is  
10 placed in tran-id field 606 of the message header.  
Specifically, as one example, exclusive access of  
transaction-counter 210 of PQMVT 200 is obtained in  
order to increment the transaction-counter. The  
transaction-counter is incremented by one and then  
15 the exclusive lock is released.

Thereafter, storage for the nexus structure to  
be associated with this new transaction is allocated  
on the system executing the PQMSEND. Then, the  
address of the nexus in the message header is placed  
20 in the next-outer-nexus-address field of the nexus  
header. The address of this nexus is copied to nexus  
address field 616 of the local message header  
(M\_HDR), STEP 1630.

Next, the remainder of the header of the newly  
25 allocated nexus is filled in, as indicated herein,  
STEP 1632. For example, tran-id 606 of M\_HDR is  
copied to next-outer-tran-id 712 of nexus 700, limb #  
610 is copied to next-outer-limb number 716, the  
queue specified on the REPLYTO() parameter is copied  
30 to queue name for collected results 718, and the  
count of the Q\_Name\_List is copied to entries needed

00585565-060700

- 40 -

to complete field 710. The replyto sysid/queue field 614 of the M\_HDR is copied to the next-outer-queue name field 714.

In addition to the above, transaction-counter 210 of PQMVT 200 is copied to tran-id 606 of M\_HDR and the queue specified on the REPLYTO() parameter is copied to replyto sysid/queue field 614 of the M\_HDR, STEP 1634. If a send to multiple targets (Q\_name\_list has more than one element) is processed as the result of a received message that is not associated with a nexus, a new nexus is created. The first message sent will have limb number one in its header. Subsequent limb-numbers for the remaining messages will be assigned sequentially. The number of limbs field of the nexus header will be set to the number of limbs used by this PQMSEND call, STEP 1636.

Subsequent to setting up the nexus header and local message header for the new transaction, the process for sending the message to the queue takes place. In one example, before the message can be sent, however, M\_HDR is copied to the message header, STEP 1637, and the queue to receive the message is identified, STEP 1638. In one example, in order to identify the position of the message in the parallel transaction, a limb number is placed in the message header copy, M\_HDR, indicating the current limb being worked on, STEP 1640. Next, the head of Q\_Name\_List is removed, STEP 1642.

Thereafter, a determination is made as to  
30 whether the queue is on the local system that issued  
POMSEND or whether it is on a remote system, INQUIRY

-47-

5

10

20

25

30

-42-

Returning to INQUIRY 1646 (FIG. 16c), if, however, the queue is not on the local system, then the message needs to be sent to the queue manager on the remote system to place the message in the queue.

5 In order to accomplish this, in one example, initially, storage is allocated for use by XCF in transferring the message to the remote system, STEP 1670 (FIG. 16e). A buffer requesting a simple PQMSEND sending the message to the identified queue

10 is created and the message is copied from the buffer identified by `buffer_ptr_list` to the newly created buffer. Thereafter, via the use of the `IXCMSGO` macro, the buffer carrying the message is sent to `PQMXXM` (a XCF exit) for calling of `PQMSEND` on the

15 identified system, STEP 1674. (`IXCMSGO` is described in detail in "MVS/ESA: Programming: Sysplex Services Reference MVS/ESA System Product: JES2 Version 5 JES3 Version 5," IBM Publication Number GC28-1496-02, June 1995, which is hereby incorporated

20 herein by reference in its entirety.) In one example, `PQMXXM` allocates storage on the remote system for the message and then the message is copied from XCF and placed in the storage. Thereafter, `PQMSEND` is called via `pqmsend()` and `buffer_ptr_list`

25 is set to the address of the allocated storage. After sending the message to the remote system, processing of `PQMSEND` then continues, INQUIRY 1664.

Returning to INQUIRY 1626 (FIG. 16b), if the specified option on `PQMSEND` is `REPLY`, INQUIRY 1679,

30 indicating that a transaction has already been started and that this is completing processing for this limb of the transaction, then the name of the target queue to receive the message is extracted from

09589566-060700



-43-

replyto sysid/queue field 614 of M\_HDR, STEP 1680. Next, a determination is made in the manner described above as to whether the extracted target queue is on the local system or a remote system, INQUIRY 1681.

5           If the target queue is on the local system, then the address of the nexus for this transaction is retrieved from the nexus address field of M\_HDR thus, enabling the nexus to be located, STEP 1682. Next, the limb # is extracted from M\_HDR and the contents  
10 of the message are copied to the nexus entry (or slot) corresponding to the extracted limb #, STEP 1683. For example, if the limb # is 5, then the message is placed as the fifth entry of the nexus. Thereafter, entries needed to complete field 710 of  
15 nexus header 702 is decremented by one, STEP 1684.

When the entries needed to complete field is greater than zero indicating that more messages are to be collected and placed in the nexus before being sent to the target queue, INQUIRY 1685, then a  
20 Program Return to the user's address space takes place, STEP 1686. However, if all of the entries have been filled in the nexus structure, INQUIRY 1685, then the messages are collected and sent to the target queue as one long message, as described in  
25 detail below.

Initially, storage is allocated for a completion message to be sent to the queue identified in the next-outer-queue-name field of the nexus header 714 and a message header is created for the new  
30 completion message, STEP 1687 (FIG. 16f). In one embodiment, this message header has the same fields

09589566-060700

as the message header of message 600. Once the message header is created, next-outer-tran-id 712 is copied from nexus header 702 and placed into the tran-id field of the new message header, and next-outer-limb number field 716 is copied to the limb # field of the new message header, STEP 1688. The next-outer-queue name 714 of the nexus header is copied into the replyto sysid/queue name field 614 of the message header. Also, the next-outer nexus address field 715 of the nexus header is copied into nexus address field 616 of the message header. The entries of the nexus are concatenated together creating one large completion message, STEP 1689.

Subsequently, the next outer queue name field in the nexus header is extracted and the value in that field overlays the queue names specified on the Q\_Name\_List parameter of the PQMSEND ensuring that the completion message is sent to the queue specified by the next outer queue name, STEP 1690. Processing then continues, as described above, with STEP 1646.

Returning to INQUIRY 1679 (FIG. 16b), if the option specified on PQMSEND is not REPLYTO() or REPLY, then flow passes to INQUIRY 1691 (FIG. 16g). At this inquiry, a determination is made as to whether more than a simple send is occurring (i.e., more than 1 message). If the transaction identifier from M\_HDR is null and the count of the Q\_Name\_List is less than or equal to one, then there is a simple send and flow continues with STEP 1637, in which the queue to receive the message is identified. However, if the transaction id is not null or the count of the Q Name List is greater than one, indicating a fan-



-46-

instruction located within the user process. The parameters for the PQMRECV Program Call include, for instance, the Q\_NAME indicating the queue to be read, a BUFFER\_PTR indicating an address of storage in the user's private area, and a NOWAIT/WAIT option indicating whether or not the process is to wait if there is nothing in the queue. The default in this embodiment is WAIT, indicating that the process will wait for messages to be sent to the queue and will not perform other work (such as, for instance, checking other queues). One embodiment of the logic associated with PQMRECV is described in detail with reference to FIG. 18.

Initially, the Q\_NAME provided as a parameter to the Program Call is searched in PQMTBL 500, STEP 1800. If it is not found, INQUIRY 1802, then the return code is set to NOT\_FOUND and a Program Return to the caller is issued, STEP 1804. However, if the requested Q\_NAME is located in the queue table, PQMTBL then an attempt to obtain exclusive access on the queue is performed, INQUIRY 1806. In one instance, exclusive access can be performed by using a COMPARE DOUBLE and SWAP (CDS) instruction on the queue pointers. Should exclusive access be denied, then it is attempted again, INQUIRY 1806. However, if exclusive access is obtained, then the entry at the head of the queue is removed and the exclusive lock is reset, STEP 1808.

(One embodiment of the COMPARE DOUBLE and SWAP instruction, Program Call instruction and Program Return instruction are described in detail in "Enterprise Systems Architecture/390 Principles of

Described above is a message processing facility  
20 that adds capabilities to a sending function, such as  
the ability to choose a default target for a message,  
initiate parallel execution, add additional processes  
to an already parallel execution and determine that a  
request has completed and perform the appropriate  
25 completion actions.

When the SEND initiates a new parallel execution, it creates a collection point for the results. This collection point is called a "nexus." The nexus is created to hold the results for this unique transaction and creates a unique transaction identifier. The very first transactions or the



limb in the current network of processes for this problem. In other words, the length of the list in the nexus is equal to the width of the parallelism in the network of tasks. This width or maximum limb number is carried in the control information in the nexus and is used determine completion. A limb may execute another multicast, effectively widening the parallelism and adding limbs to the topology of the network of processes for this transaction. Added send function detects this widening, provides unique limb numbers for the new limbs and adds entries to the nexus list.

The above functions can be used recursively. In other words, a new request and its corresponding

10           The mechanisms of the present invention can be  
included in one or more computer program products  
including computer useable media, in which the media  
include computer readable program code means for  
providing and facilitating the mechanisms of the  
15   present invention. The products can be included as  
part of a computer system or sold separately.

25           Although preferred embodiments have been  
depicted and described in detail herein, it will be  
apparent to those skilled in the relevant art that  
various modifications, additions, substitutions and  
the like can be made without departing from the  
30   spirit of the invention and these are therefore



-51-

considered to be within the scope of the invention as defined in the following claims.

09523136 130700

-52-

CLAIMS

What is claimed is:

- 1           1.    A method for processing messages in a  
2   computer system, comprising:
- 3                sending a message to a location designated  
4   to receive said message, said message comprising  
5   a request identifier specifying a request said  
6   message corresponds to and a sequence identifier  
7   indicating where in a processing sequence of  
8   said request said message belongs; and
- 9                retrieving said message from said location.
- 1           2.    The method of claim 1, wherein said sending  
2   comprises using a send function to send said message  
3   to said location, said send function including a  
4   parameter indicating whether said request is a new  
5   request.
- 1           3.    The method of claim 2, wherein said  
2   parameter indicates said request is a new request and  
3   said parameter further comprises a target parameter  
4   to receive results for said new request when  
5   processing for said new request is completed.

09589566.050700



10 means for retrieving said message from said  
11 location.

1           9. The system of claim 8, wherein said  
2     parameter indicates said request is a new request and  
3     said parameter further comprises a target parameter  
4     to receive results for said new request when  
5     processing for said new request is completed.

-55-

1           10. The system of claim 7, wherein said means  
2     for retrieving comprises:

3                 means for removing said request identifier  
4             and said sequence identifier prior to retrieval  
5             of said message.

1           11. The system of claim 7, further comprising  
2     means for determining completion of said processing  
3     sequence.

1           12. The system of claim 7, wherein said means  
2     for retrieving comprises means for retrieving said  
3     message by a process and said system further  
4     comprises said process sending said message to one or  
5     more other processes dynamically determined, by said  
6     process, to receive said message.

\* \* \* \* \*

007090-060700

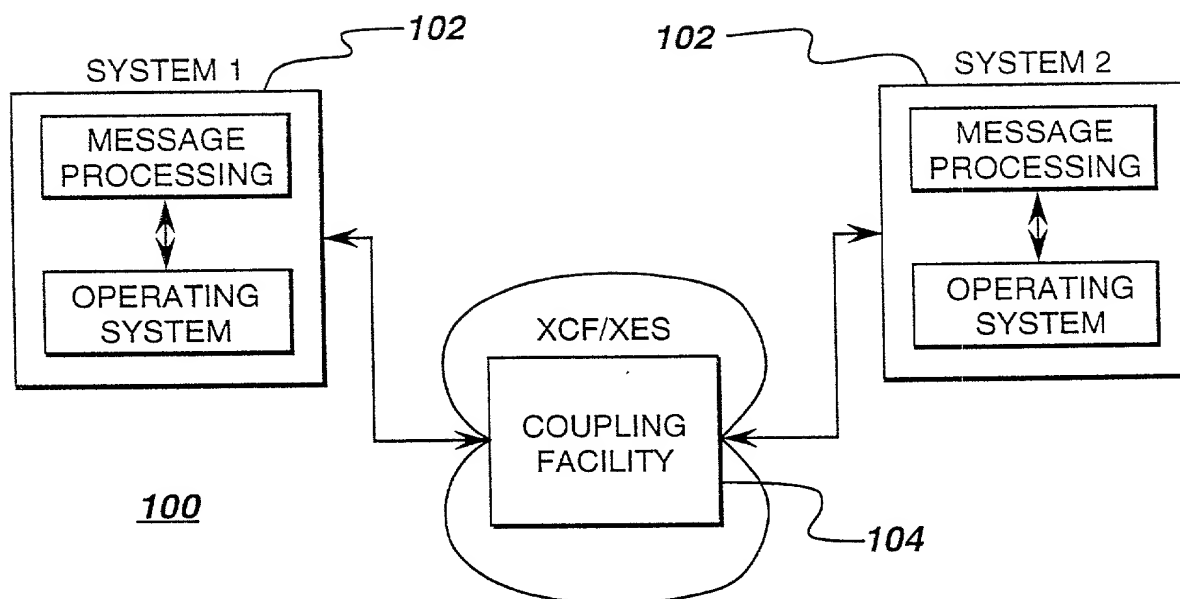
-56-

METHOD AND SYSTEM FOR PROCESSING MESSAGES  
IN A DISTRIBUTED COMPUTING ENVIRONMENT

ABSTRACT OF THE DISCLOSURE

A message processing facility provides a send function for sending messages to designated locations. The facility allows a default target for a message, initiates parallel execution, adds  
5 additional processes to an already parallel execution, determines that a problem has completed and performs the appropriate completion actions. The message processing facility enables coordination of the results of a network that has been widened in its  
10 parallelism, without creating subproblems. Processes within the parallel network do not need to be aware of the topology or the position of a given process within the topology. The width of the parallelism involved in a particular request is tracked, and  
15 therefore, the appropriate number of results expected can be determined automatically and dynamically. Therefore, despite dynamic changes in topology, completion can be determined without awareness of the processes that create the results.

09589566-060700



*fig. 1*

002090" 9956860

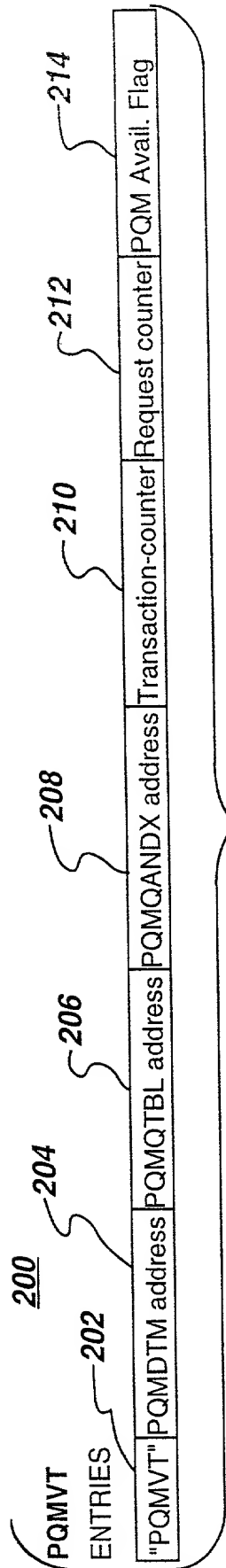


fig. 2

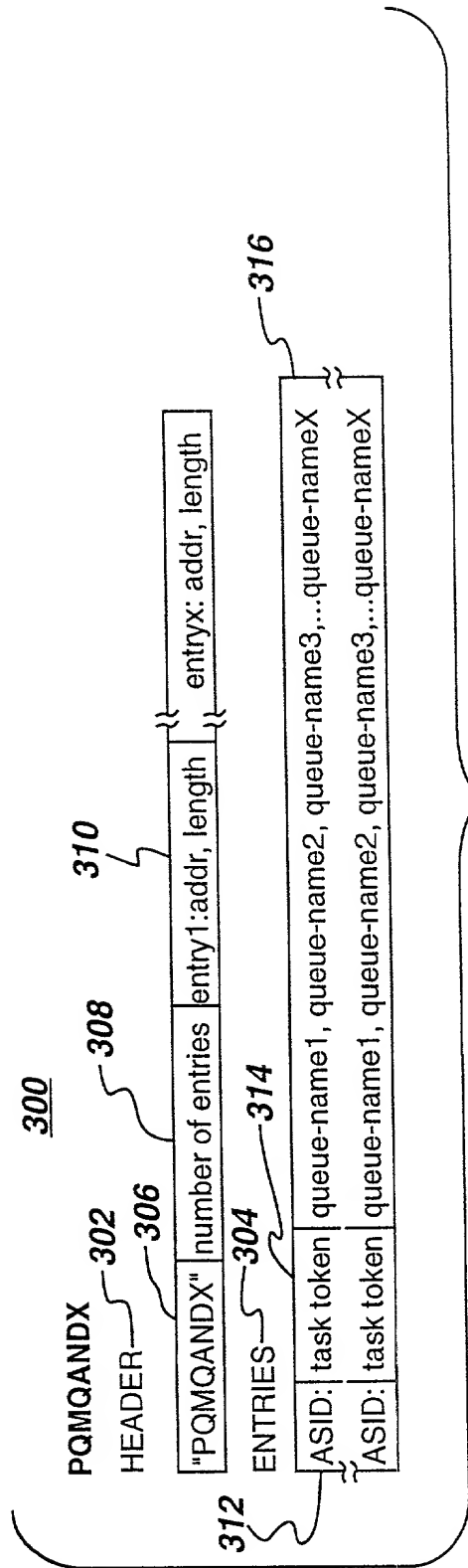
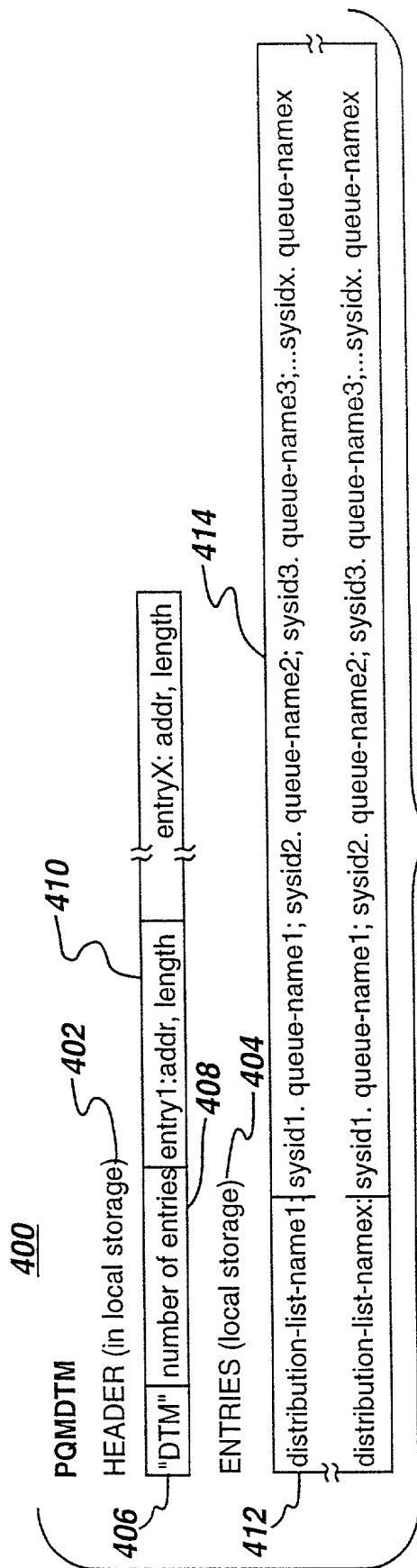
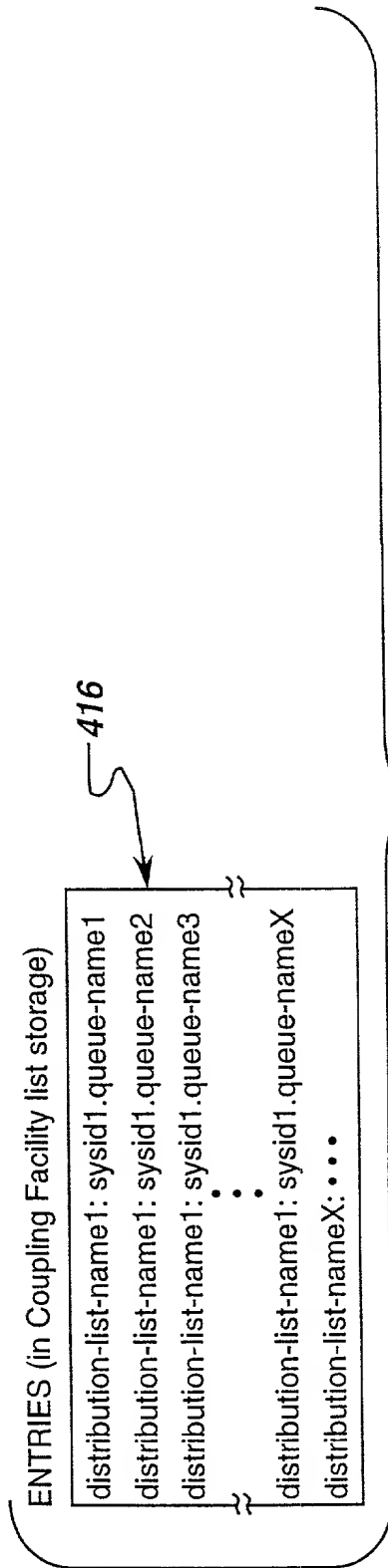


fig. 3





*fig. 4a*



*fig. 4b*

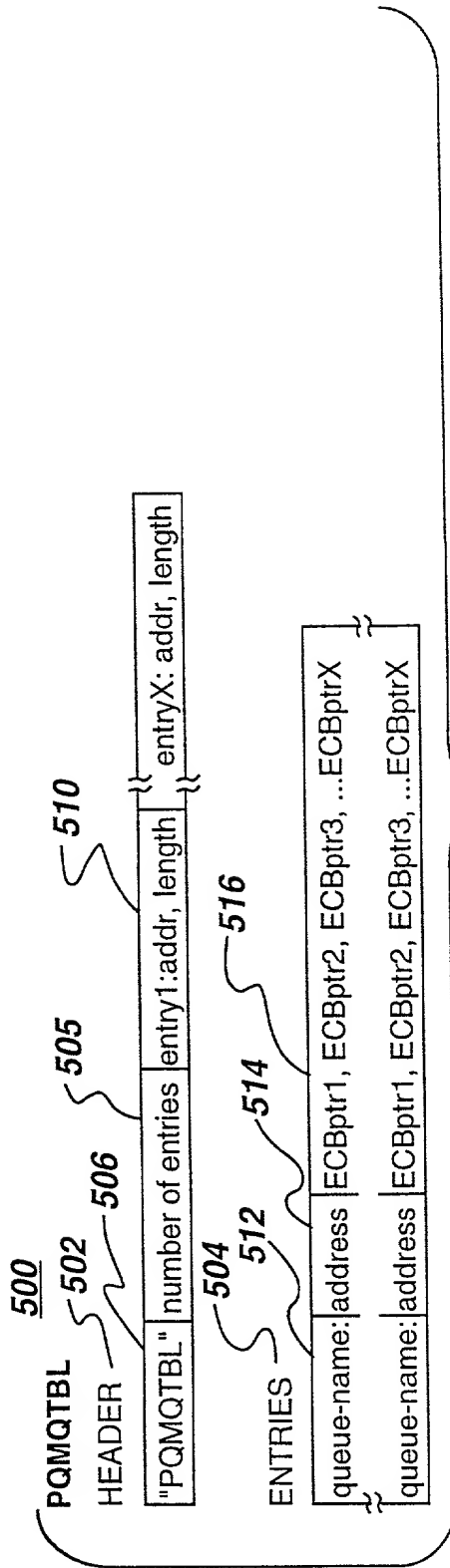


fig. 5

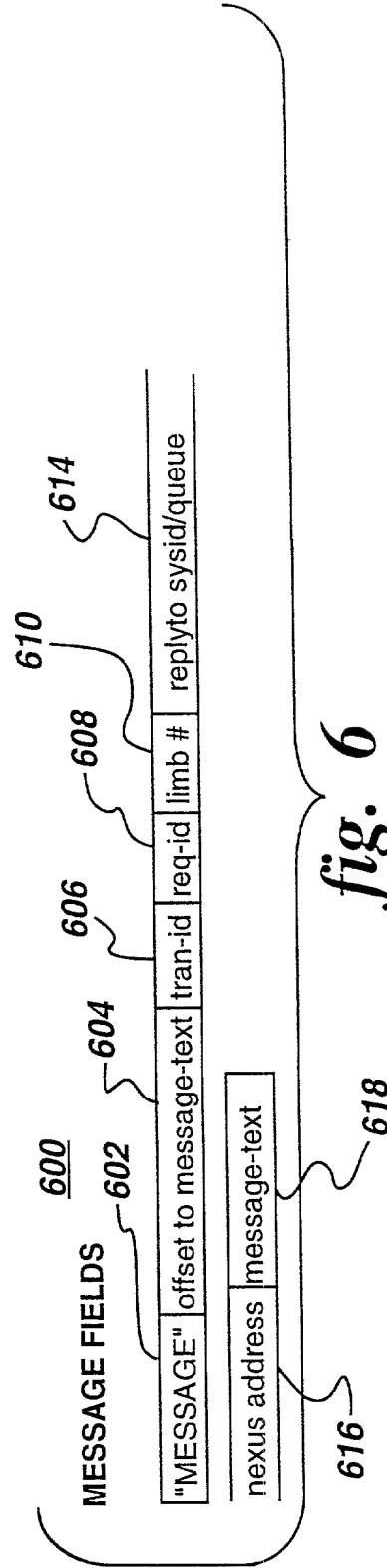
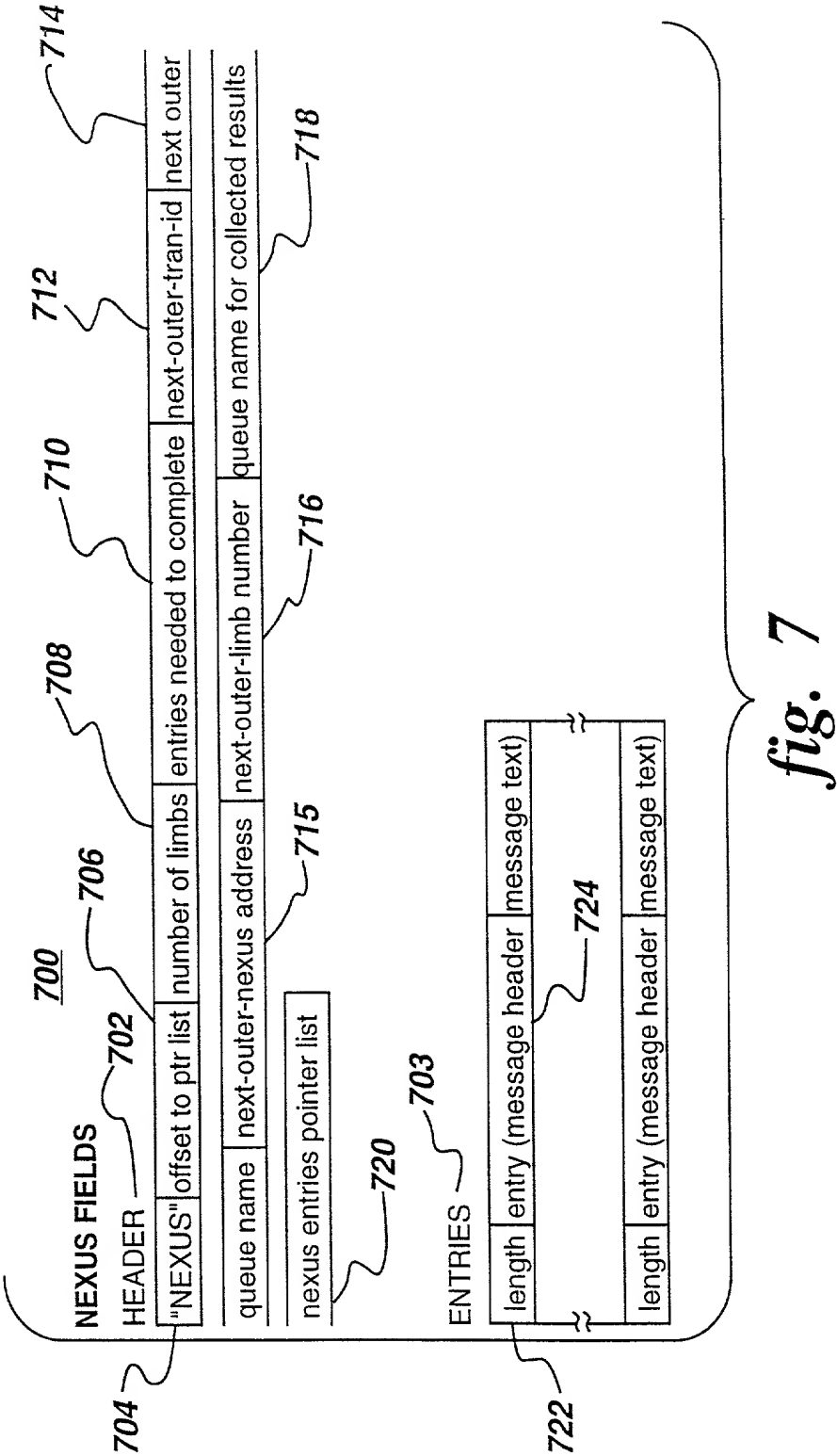
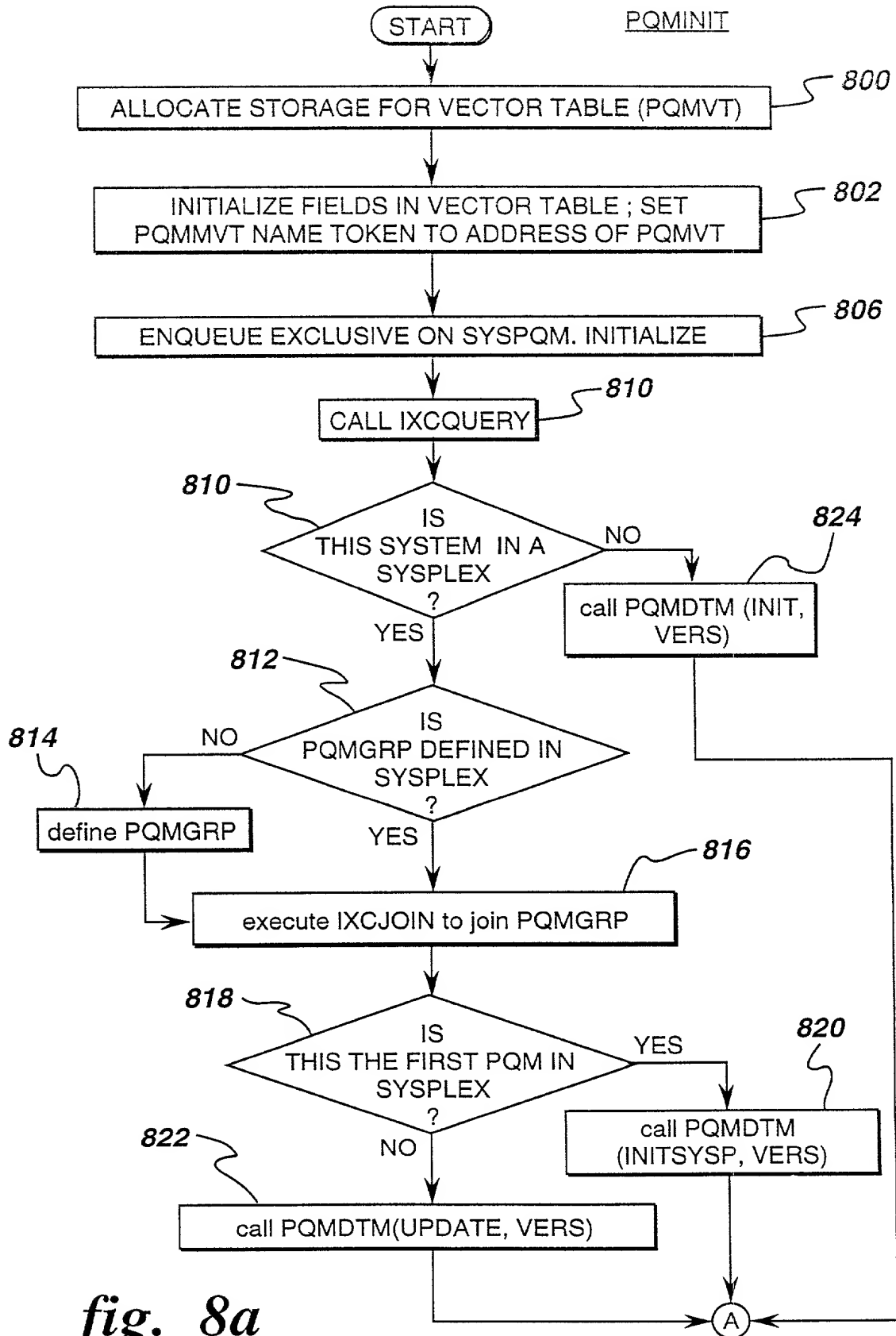
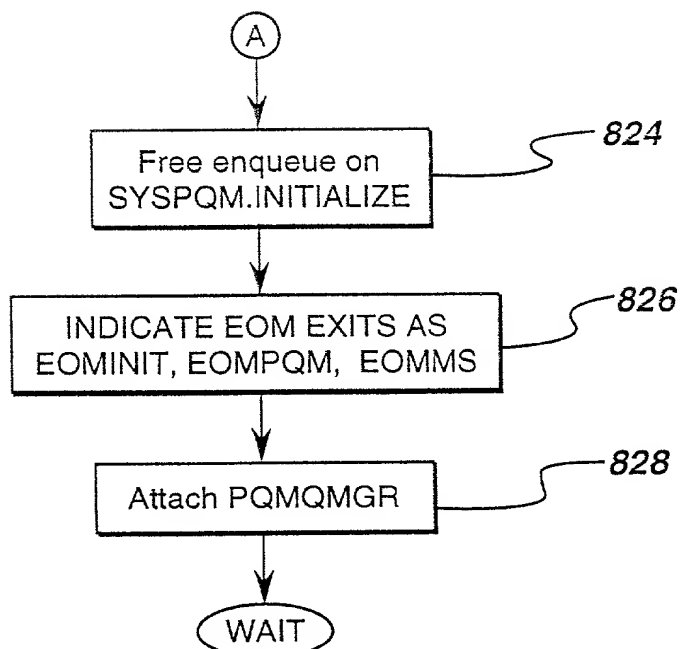
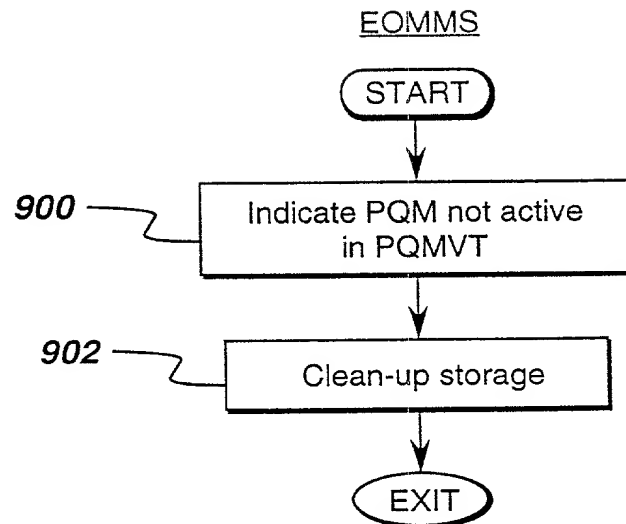
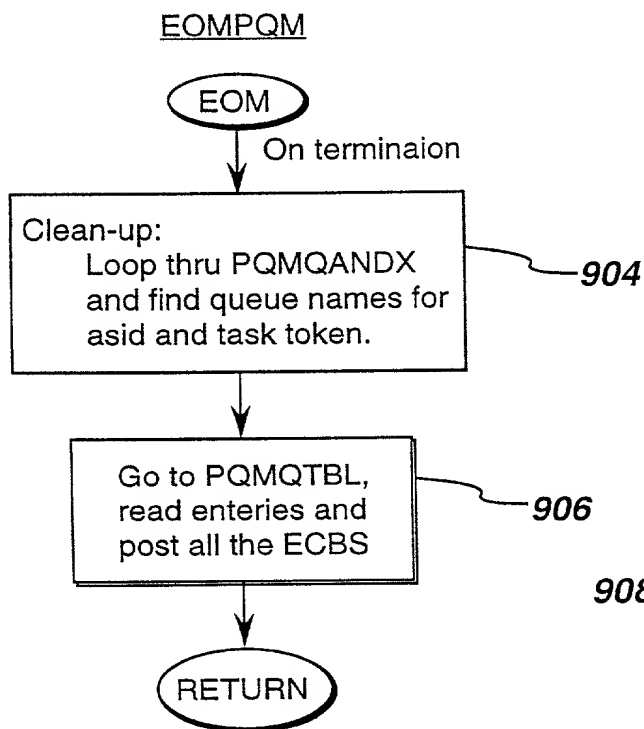
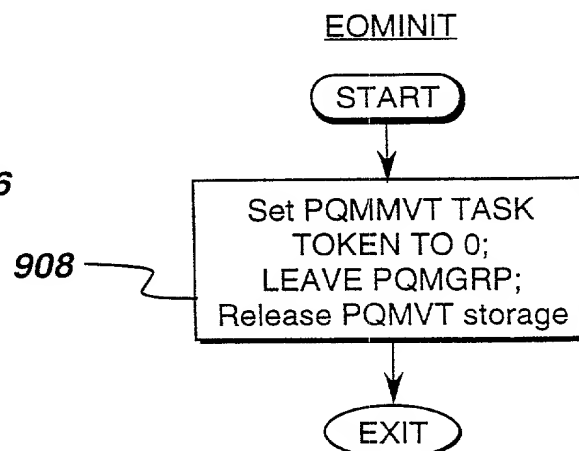


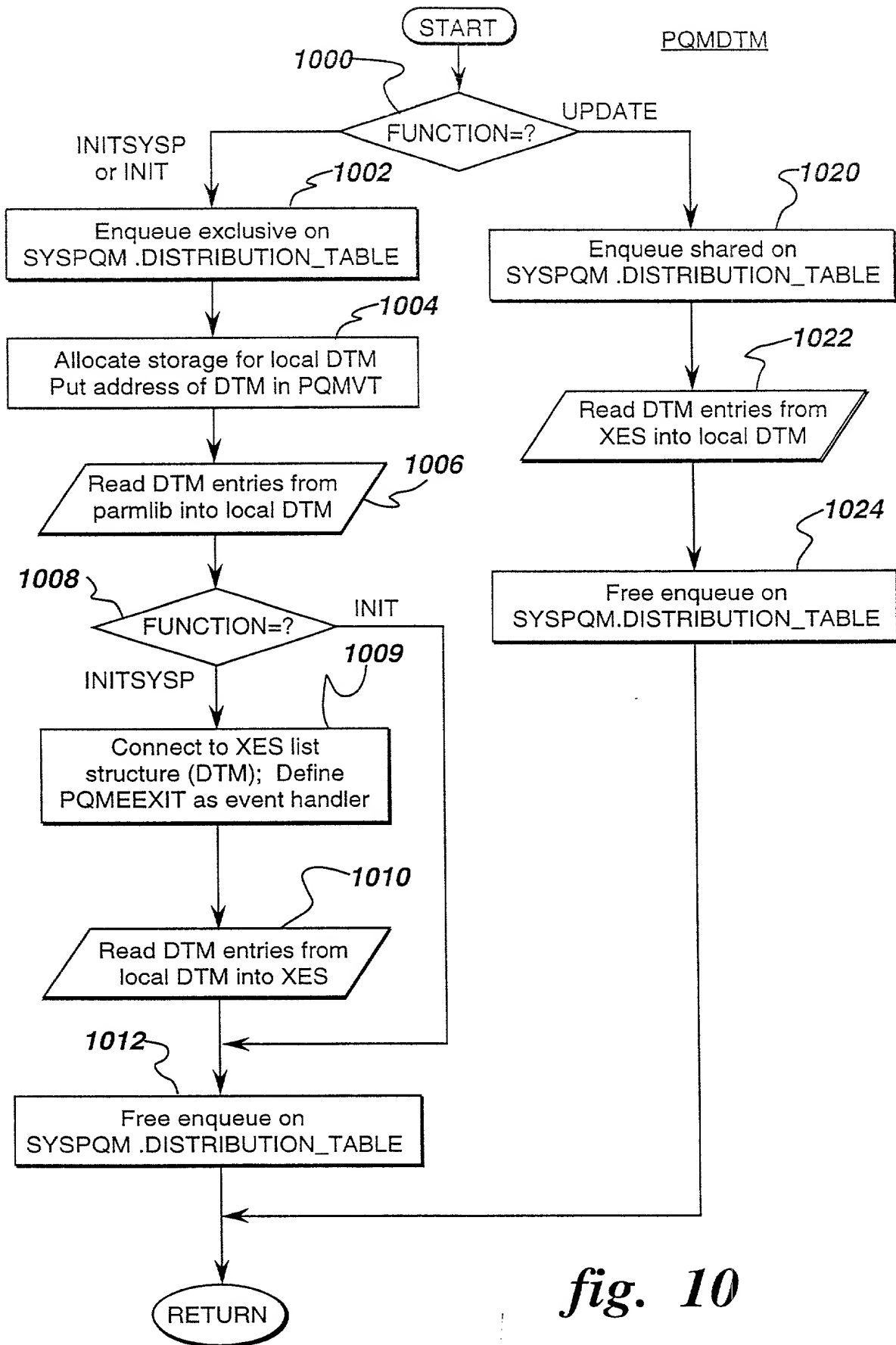
fig. 6

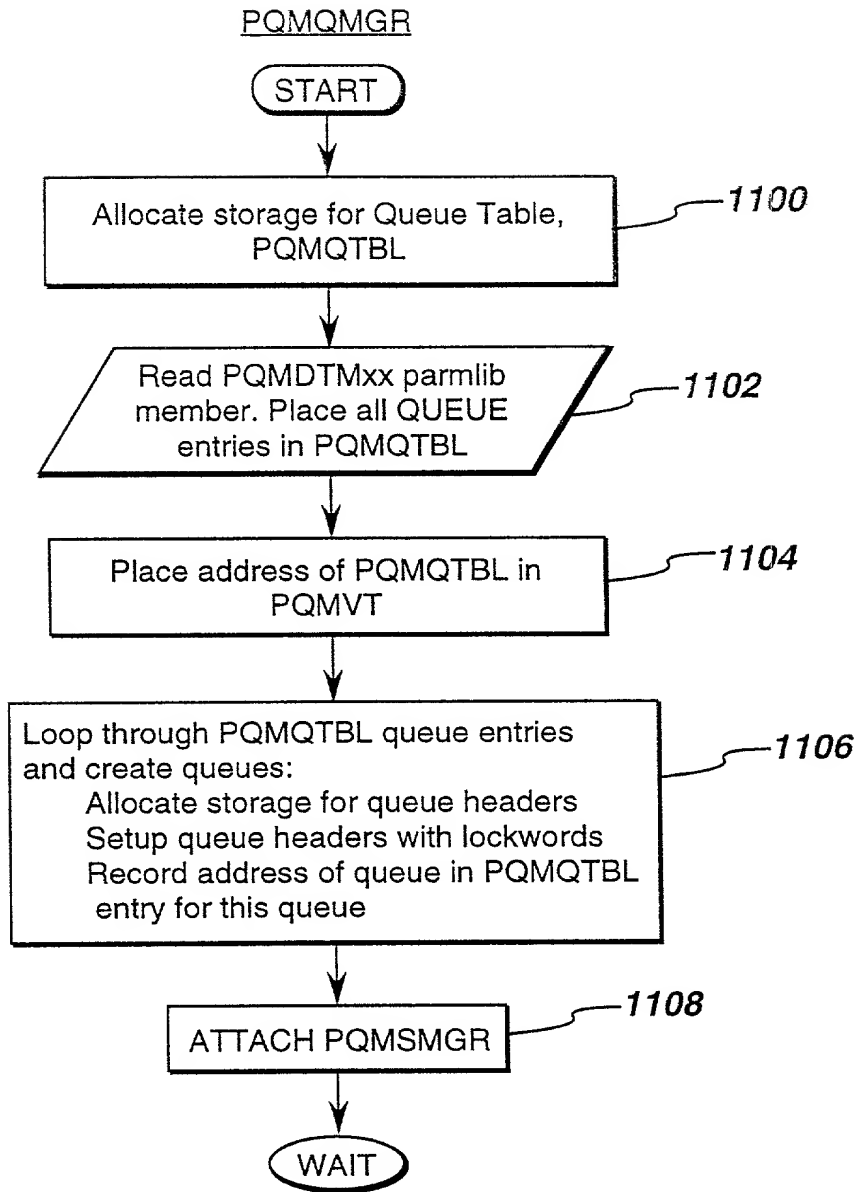


*fig. 8a*

*fig. 8b**fig. 9a**fig. 9b**fig. 9c*

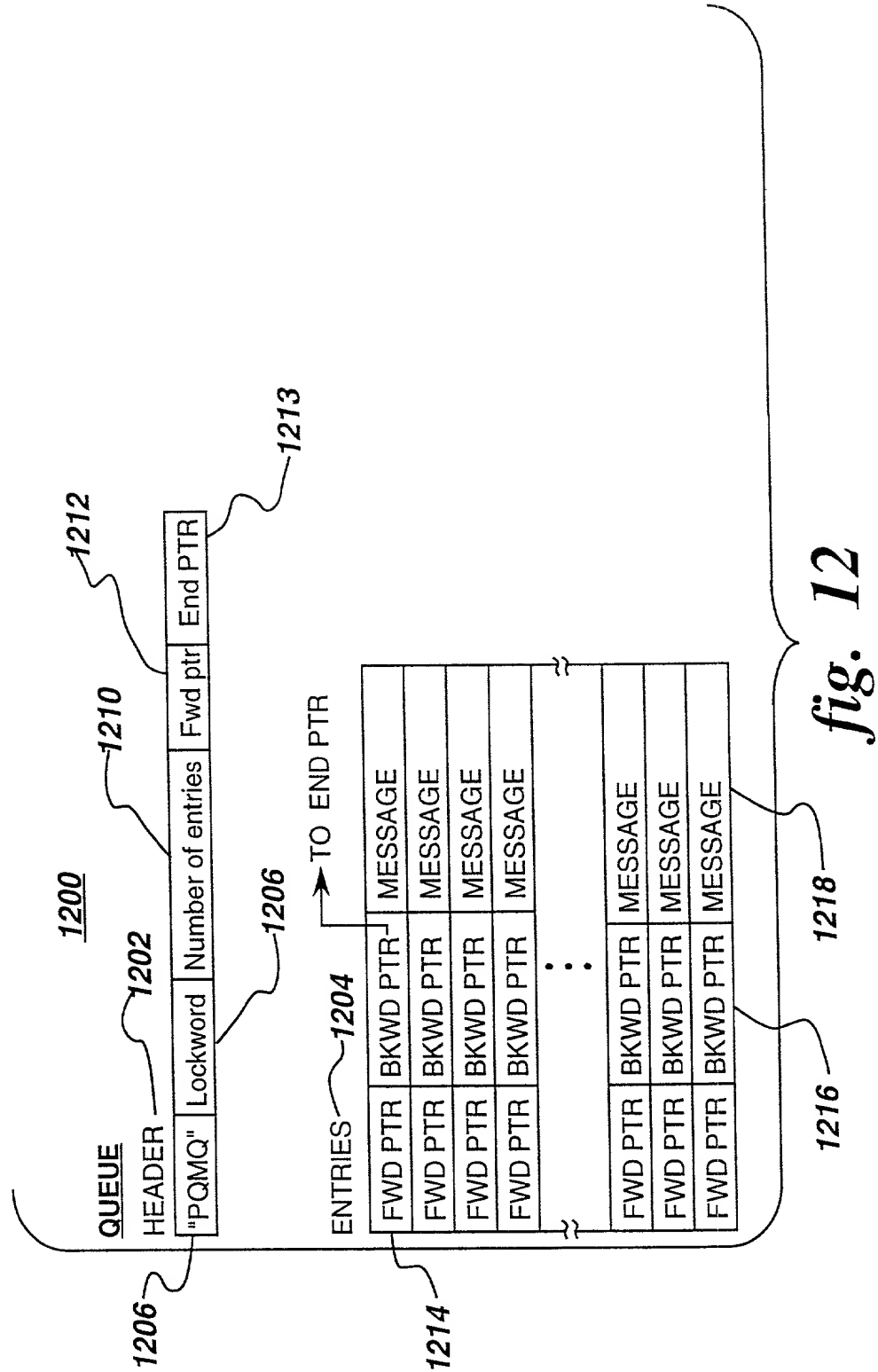
007090" 99568560



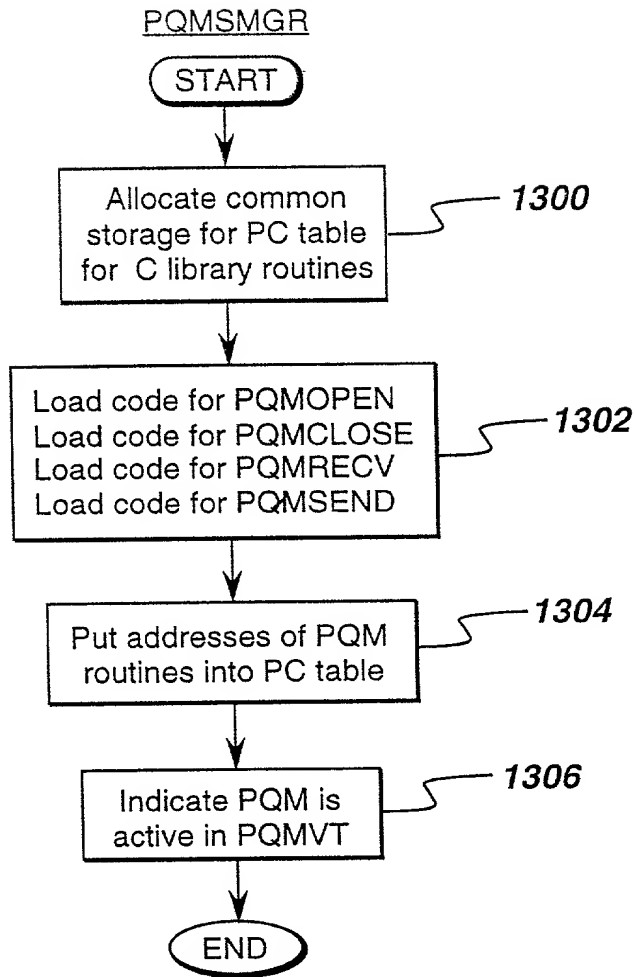


*fig. 11*

09589566-060700







*fig. 13*

05589556 " 060700

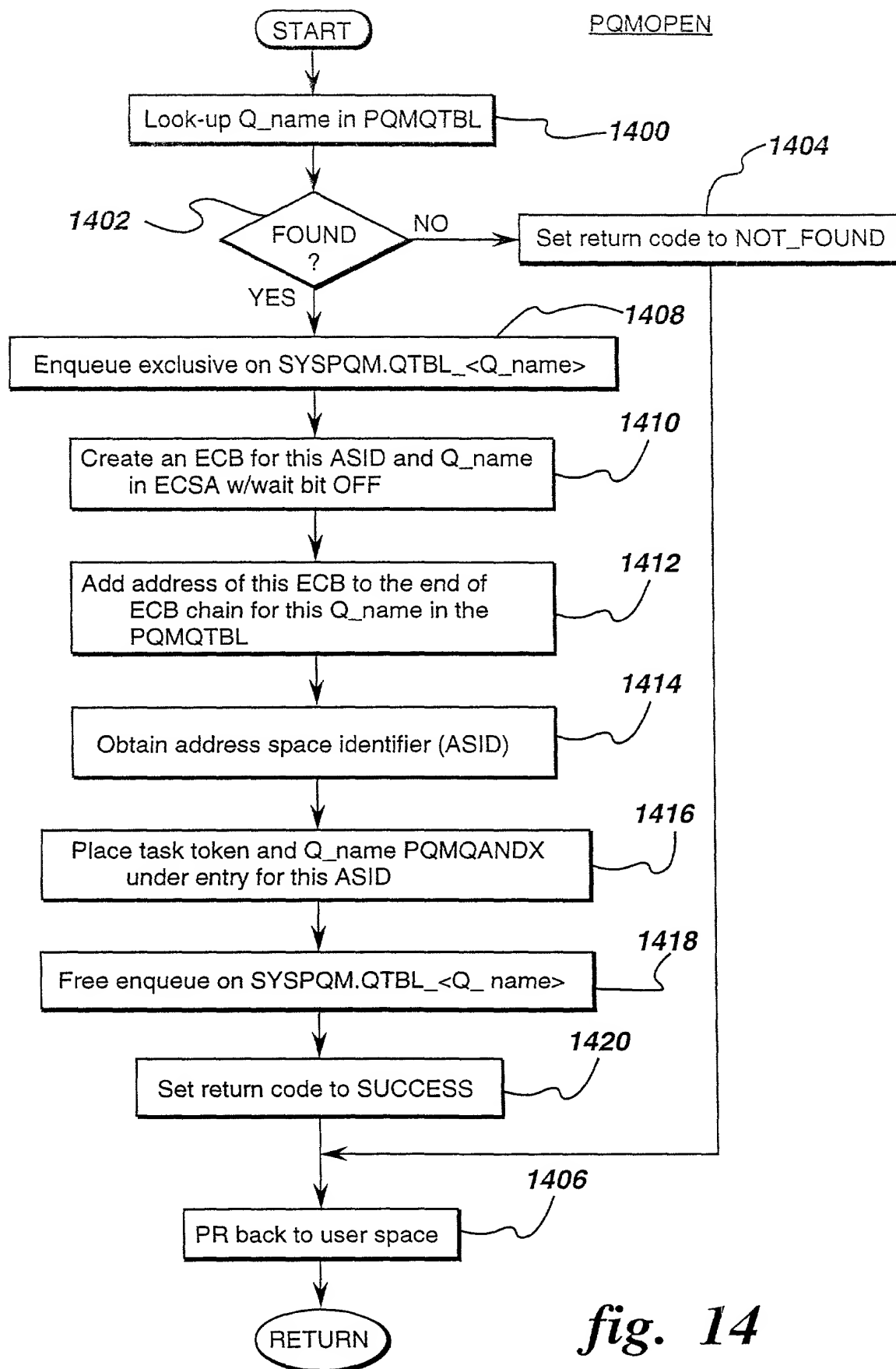


fig. 14

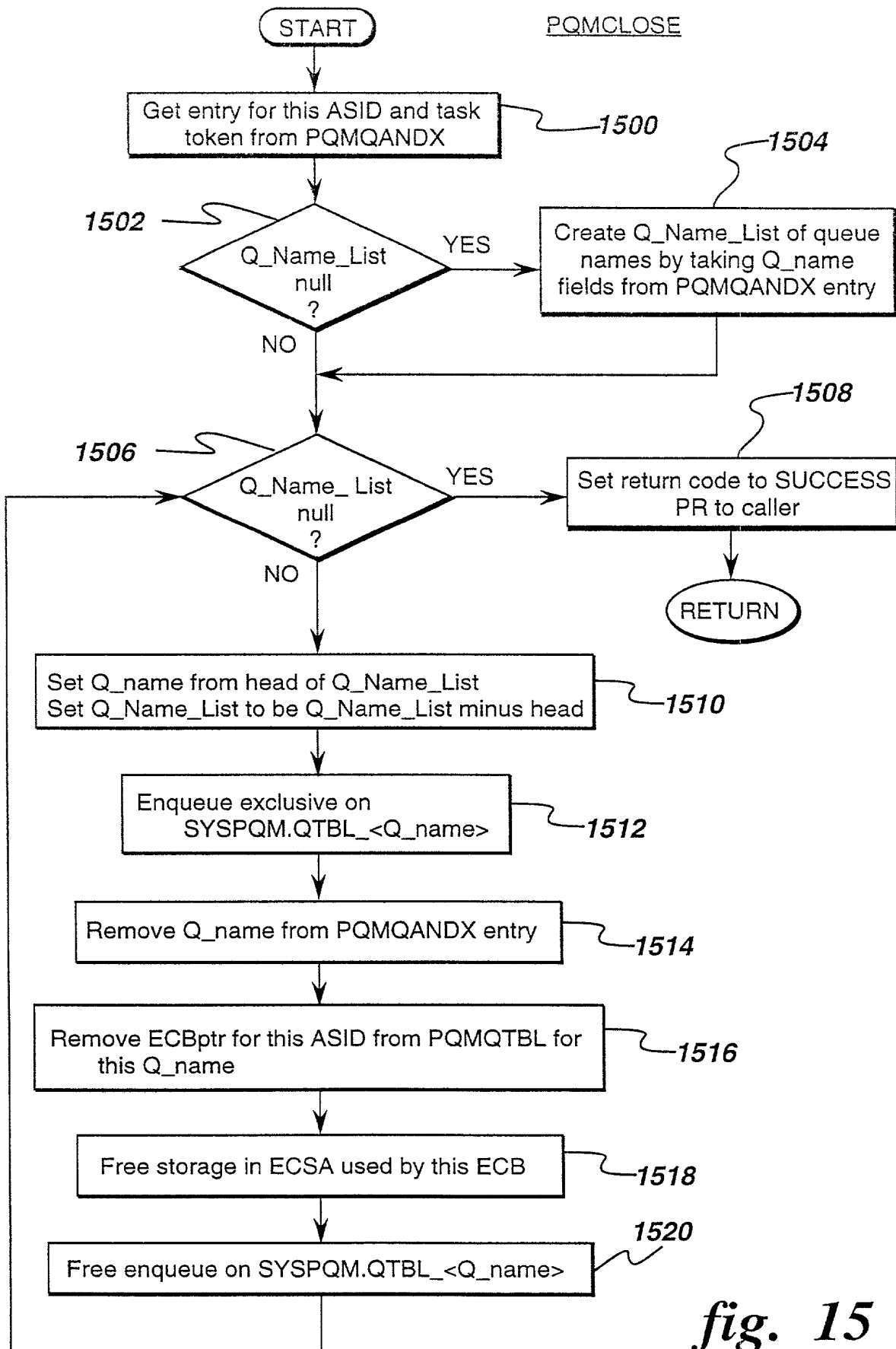
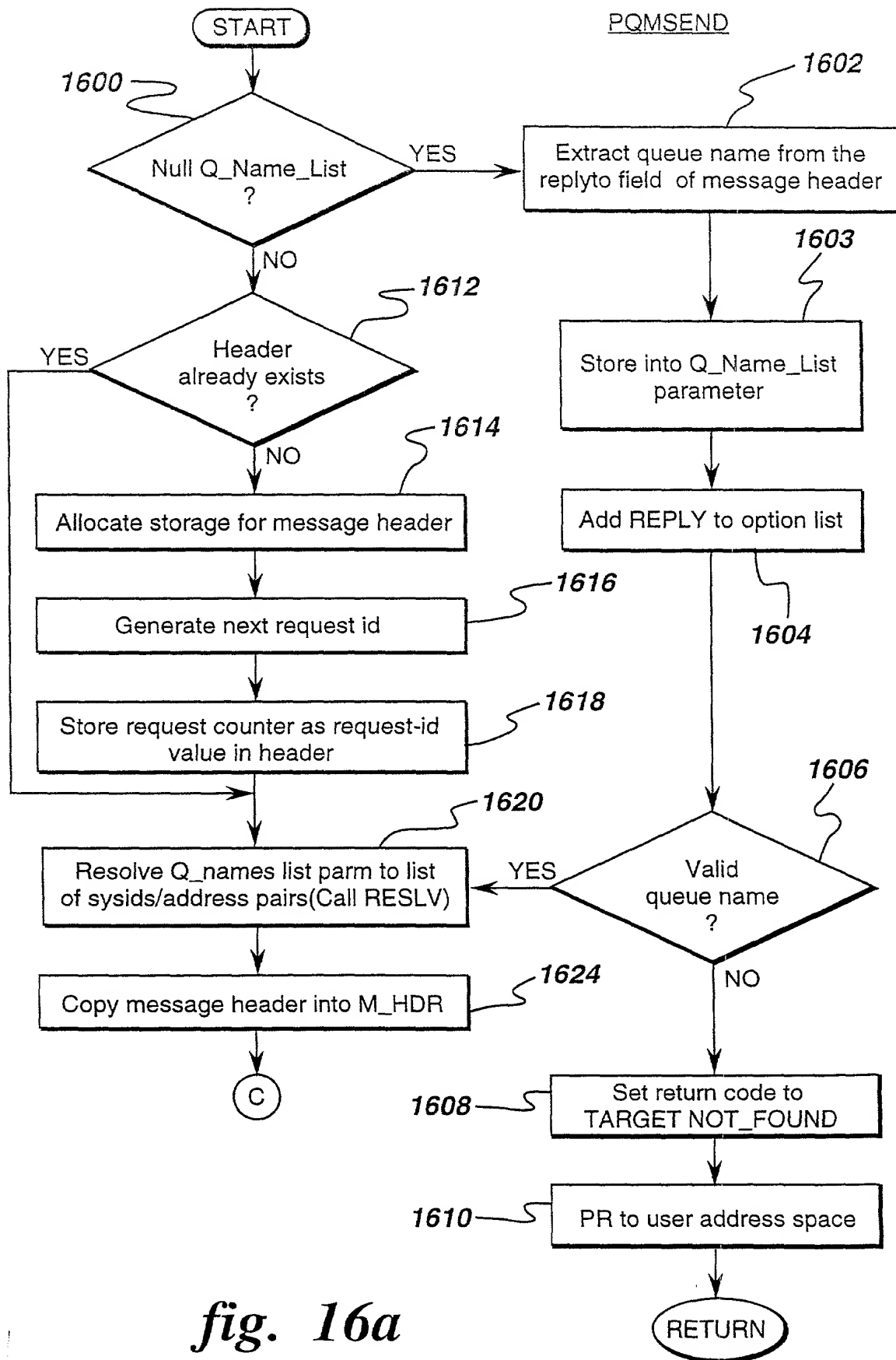
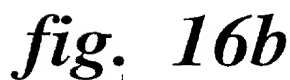
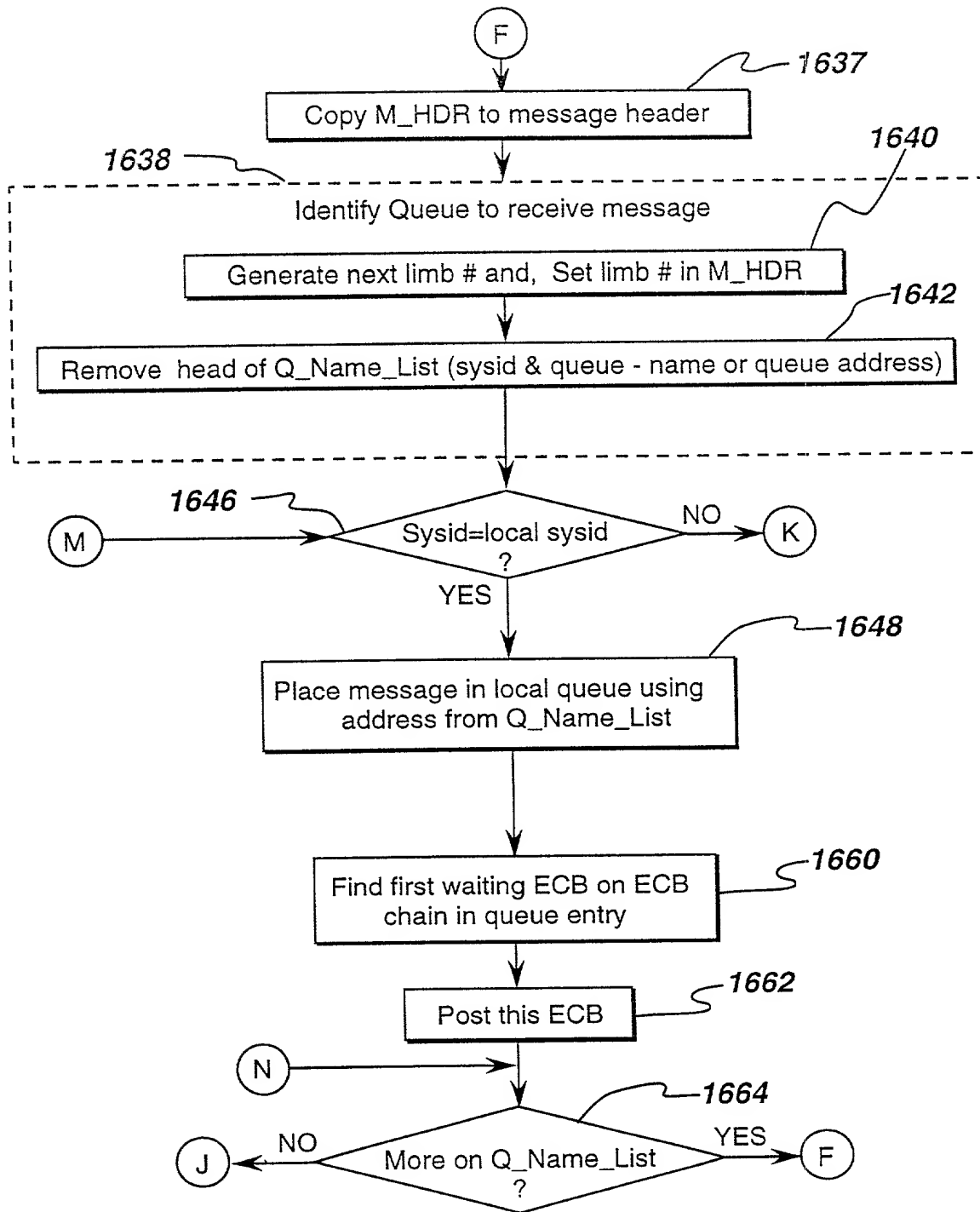


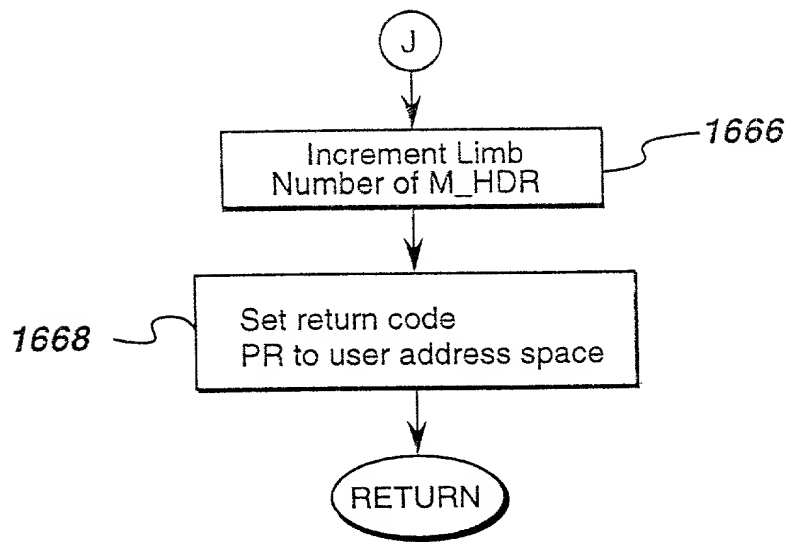
fig. 15



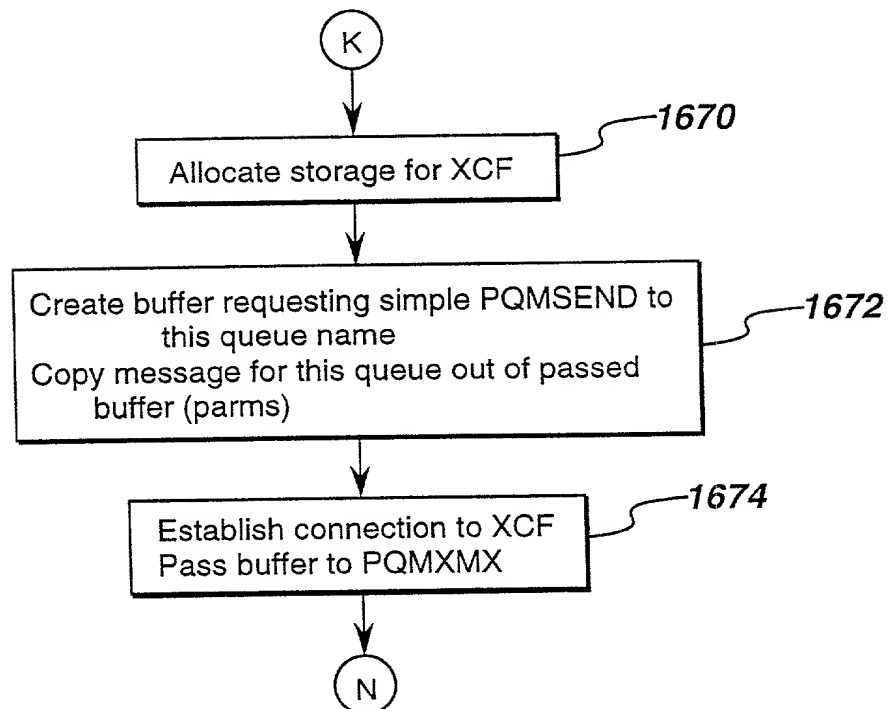




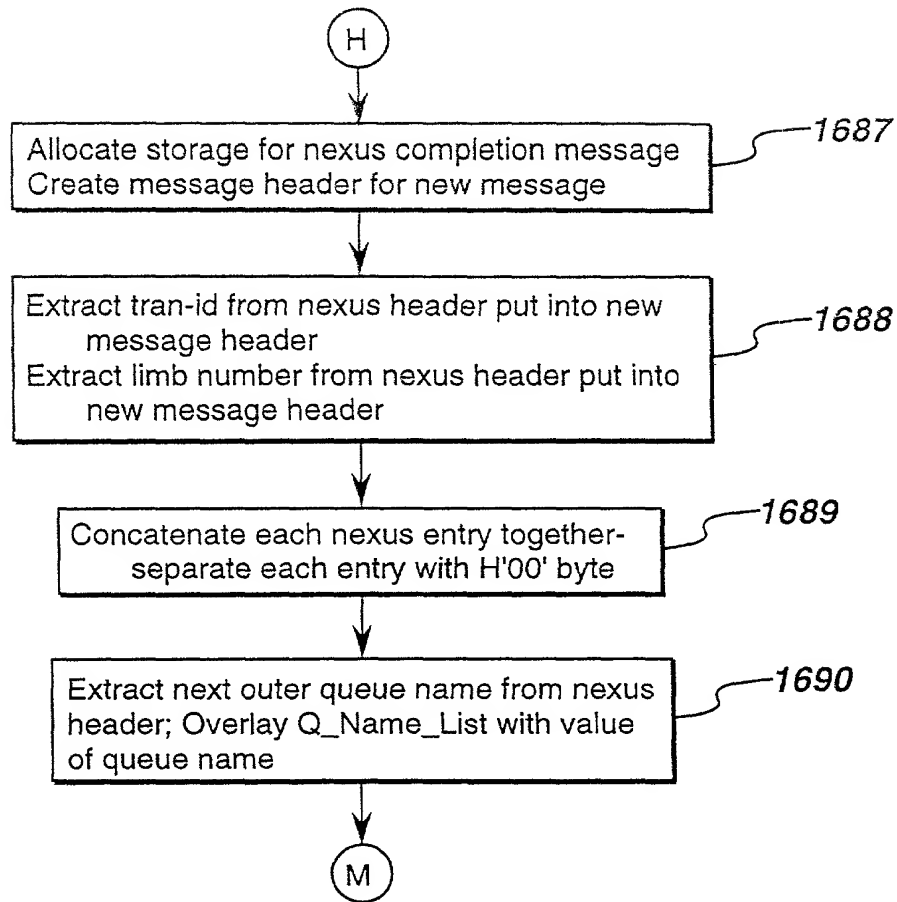
*fig. 16c*



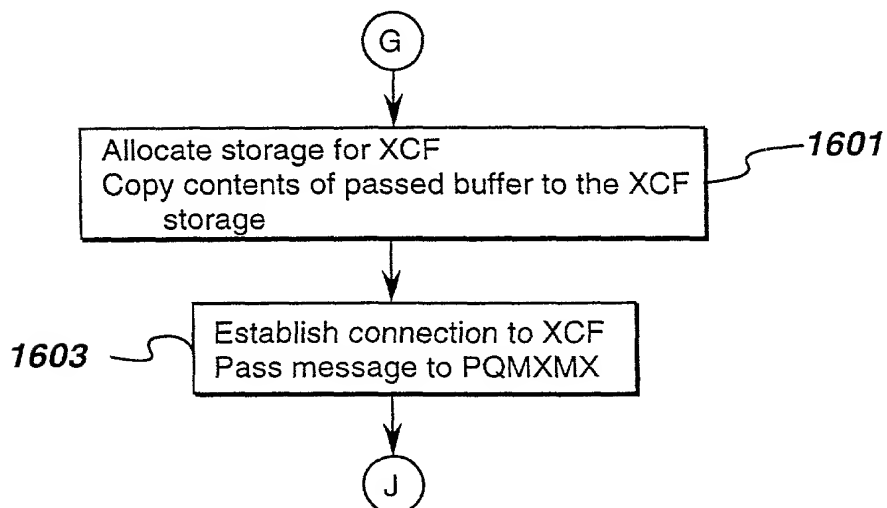
*fig. 16d*



*fig. 16e*

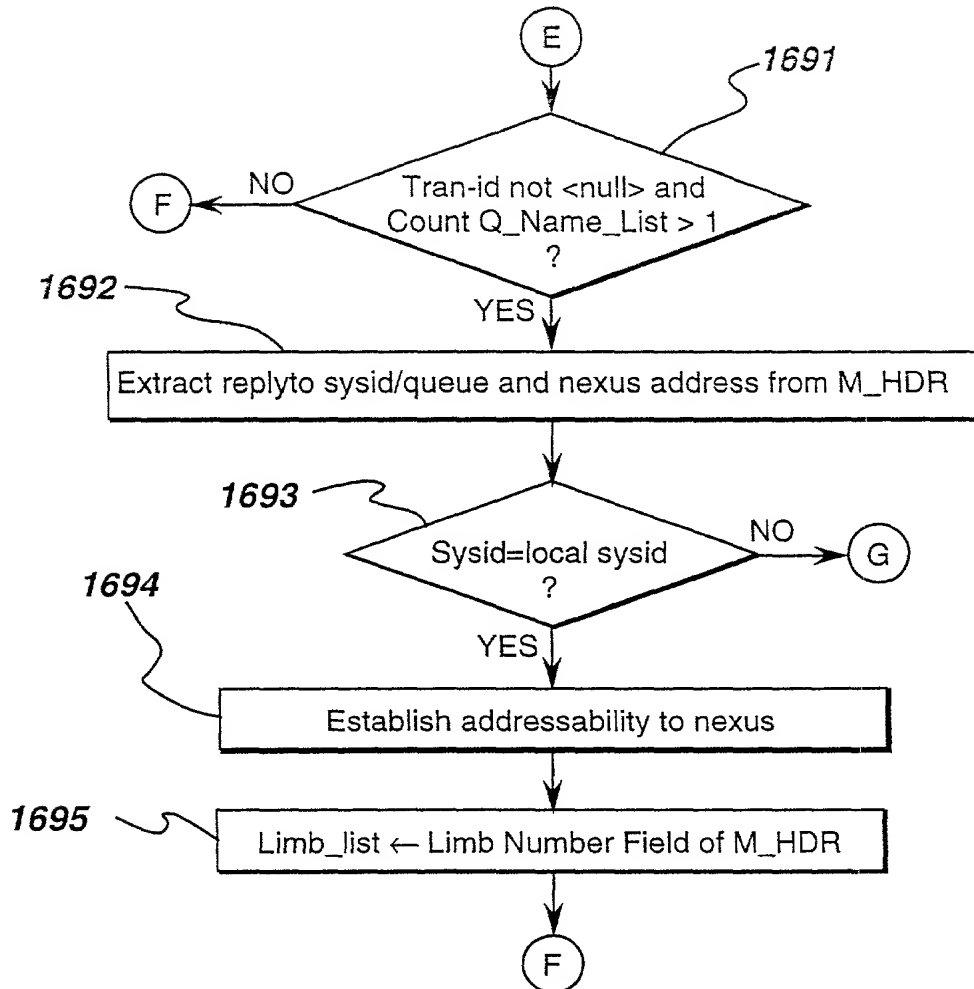


*fig. 16f*



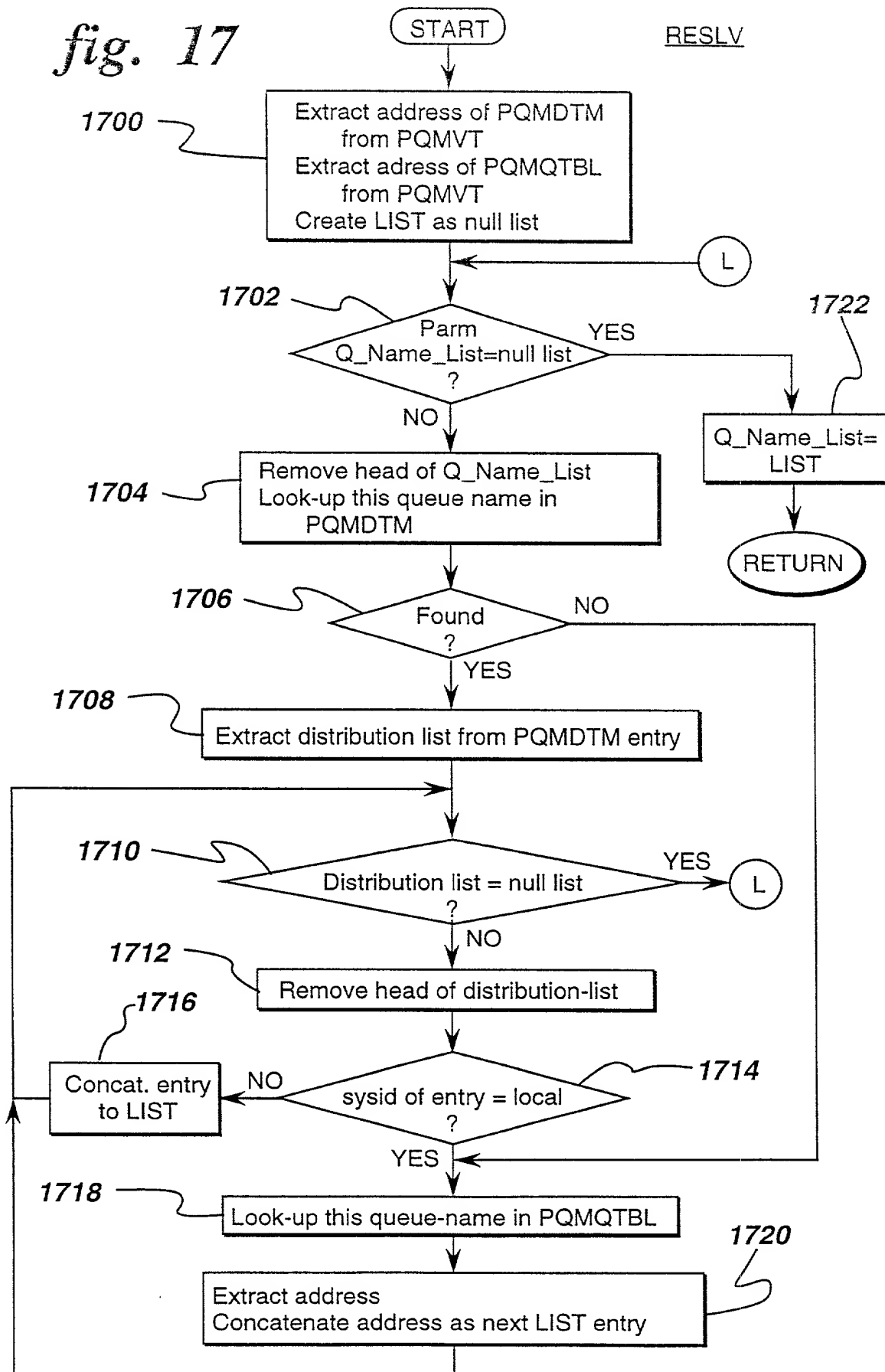
*fig. 16h*

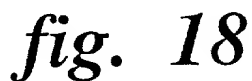




*fig. 16g*

fig. 17





## DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name. I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD AND SYSTEM FOR PROCESSING MESSAGES  
IN A DISTRIBUTED COMPUTING ENVIRONMENT

the specification of which (check one)

X is attached hereto.

           was filed on                                  as United States  
Application Number or PCT International Application  
Number                                  and was amended on  
                                .

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above. I acknowledge the duty to disclose to the U.S. Patent and Trademark Office all information known to me to be material to patentability as defined in 37 CFR § 1.56.

I hereby claim foreign priority benefits under 35 U.S.C. § 119(a)-(d) or § 365(b) of any foreign application(s) for patent or inventor's certificate, or § 365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below any foreign application for patent or inventor's certificate, or PCT International application having a filing date before that of the application on which priority is claimed.

## Priority Claimed

<u>                                </u> (Number)	<u>                                </u> (Country)	<u>                                </u> (Day/Month/Year Filed)	<u>      </u> Yes <u>      </u> No
<u>                                </u> (Number)	<u>                                </u> (Country)	<u>                                </u> (Day/Month/Year Filed)	<u>      </u> Yes <u>      </u> No

I hereby claim the benefit under 35 U.S.C. § 119(e) of any United States provisional application(s) listed below.

<u>                                </u> (Application Number)	<u>                                </u> (Filing Date)
<u>                                </u> (Application Number)	<u>                                </u> (Filing Date)

I hereby claim the benefit under 35 U.S.C. § 120 of any United States application(s), or § 365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of 35 U.S.C. § 112, I acknowledge the duty to disclose material information as defined in 37 CFR §1.56(a) which occurred between the filing date of the prior application and the national or PCT International filing date of this application:

(Appl. Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)
--------------------	---------------	---

(Appl. Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)
--------------------	---------------	---

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

Lynn L. Augspurger, Esq.	Reg. No. 24,227
Lawrence D. Cutter, Esq.	Reg. No. 28,501
Marc A. Ehrlich, Esq.	Reg. No. 39,966
Bernard M. Goldman, Esq.	Reg. No. 17,959
Floyd A. Gonzalez, Esq.	Reg. No. 26,732
William A. Kinnaman, Jr., Esq	Reg. No. 27,650
Lily Neff, Esq.	Reg. No. 38,254
John A. Jordan, Esq.	Reg. No. 24,655
Jeff Rothenberg, Esq.	Reg. No. 26,429
Kevin P. Radigan, Esq.	Reg. No. 31,789
Blanche E. Schiller, Esq.	Reg. No. 35,670

Christopher A. Hughes, Esq., Reg. No. 26,914; Edward A. Pennington, Reg. No. 32,588  
John E. Hoel, Esq., Reg. No. 26,279; Joseph C. Redmond, Jr., Reg. No. 18,753

Send Correspondence to:

Blanche E. Schiller, Esq.  
HESLIN & ROTHENBERG, P.C.  
5 Columbia Circle  
Albany, New York 12203-5160  
Telephone: (518) 452-5600  
Facsimile: (518) 452-5579

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

ADDED PAGE(S) TO COMBINED DECLARATION AND POWER OF ATTORNEY  
FOR SIGNATURE BY FIRST AND SUBSEQUENT INVENTORS

Full Name of sole or first inventor: Scott Andrew Fagen

Signature:  Date: 10/10/96

Residence: #2 Earl Court, Poughkeepsie, New York 12603

Citizenship: United States of America

Post Office Address: #2 Earl Court, Poughkeepsie, New York 12603

Full Name of second joint inventor: Richard Charles Williams

Signature: \_\_\_\_\_ Date: \_\_\_\_\_

Residence: #2 Jansen Road, New Paltz, New York 12561

Citizenship: United States of America

Post Office Address: #2 Jansen Road, New Paltz, New York 12561

Full Name of third joint inventor:

Signature: \_\_\_\_\_ Date: \_\_\_\_\_

Residence:

Citizenship:

Post Office Address:

Full Name of fourth joint inventor:

Signature: \_\_\_\_\_ Date: \_\_\_\_\_

Residence:

Citizenship:

Post Office Address:

002090-9955560

EXPRESS MAIL LABEL NO. \_\_\_\_\_

DOCKET NO. PO9-95-064  
PAGE 1 OF 3

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name. I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD AND SYSTEM FOR PROCESSING MESSAGES  
IN A DISTRIBUTED COMPUTING ENVIRONMENT

the specification of which (check one)

\_\_\_\_\_ is attached hereto.

X was filed on October 11, 1996 as United States  
Application Number or PCT International Application  
Number 08/730,527 and was amended on  
\_\_\_\_\_.

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above. I acknowledge the duty to disclose to the U.S. Patent and Trademark Office all information known to me to be material to patentability as defined in 37 CFR § 1.56.

I hereby claim foreign priority benefits under 35 U.S.C. § 119(a)-(d) or § 365(b) of any foreign application(s) for patent or inventor's certificate, or § 365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below any foreign application for patent or inventor's certificate, or PCT International application having a filing date before that of the application on which priority is claimed.

Priority Claimed

_____ (Number)	_____ (Country)	_____ (Day/Month/Year Filed)	____ Yes	____ No
_____ (Number)	_____ (Country)	_____ (Day/Month/Year Filed)	____ Yes	____ No

I hereby claim the benefit under 35 U.S.C. §119(e) of any United States provisional application(s) listed below.

_____ (Application Number)	_____ (Filing Date)
_____ (Application Number)	_____ (Filing Date)

05589566-060700

I hereby claim the benefit under 35 U.S.C. § 120 of any United States application(s), or § 365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of 35 U.S.C. § 112, I acknowledge the duty to disclose material information as defined in 37 CFR §1.56(a) which occurred between the filing date of the prior application and the national or PCT International filing date of this application:

(Appl. Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)
--------------------	---------------	---

(Appl. Serial No.)	(Filing Date)	(Status) (patented, pending, abandoned)
--------------------	---------------	---

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

Lynn L. Augspurger, Esq.	Reg. No. 24,227
Lawrence D. Cutter, Esq.	Reg. No. 28,501
Marc A. Ehrlich, Esq.	Reg. No. 39,966
Bernard M. Goldman, Esq.	Reg. No. 17,959
Floyd A. Gonzalez, Esq.	Reg. No. 26,732
William A. Kinnaman, Jr., Esq.	Reg. No. 27,650
Lily Neff, Esq.	Reg. No. 38,254
John A. Jordan, Esq.	Reg. No. 24,655
Jeff Rothenberg, Esq.	Reg. No. 26,429
Kevin P. Radigan, Esq.	Reg. No. 31,789
Blanche E. Schiller, Esq.	Reg. No. 35,670

Christopher A. Hughes, Esq., Reg. No. 26,914; Edward A. Pennington, Reg. No. 32,588  
John E. Hoel, Esq., Reg. No. 26,279; Joseph C. Redmond, Jr., Reg. No. 18,753

Send Correspondence to:

Blanche E. Schiller, Esq.  
HESLIN & ROTHENBERG, P.C.  
5 Columbia Circle  
Albany, New York 12203-5160  
Telephone: (518) 452-5600  
Facsimile: (518) 452-5579

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.



ADDED PAGE(S) TO COMBINED DECLARATION AND POWER OF ATTORNEY  
FOR SIGNATURE BY FIRST AND SUBSEQUENT INVENTORS

Full Name of sole or first inventor: Scott Andrew Fagen

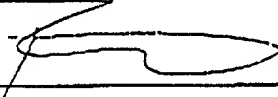
Signature: \_\_\_\_\_ Date: \_\_\_\_\_

Residence: #2 Earl Court, Poughkeepsie, New York 12603

Citizenship: United States of America

Post Office Address: #2 Earl Court, Poughkeepsie, New York 12603

Full Name of second joint inventor: Richard Charles Williams

Signature:  \_\_\_\_\_ Date: 1/2/97

Residence: #2 Jansen Road, New Paltz, New York 12561

Citizenship: United States of America

Post Office Address: #2 Jansen Road, New Paltz, New York 12561

Full Name of third joint inventor:

Signature: \_\_\_\_\_ Date: \_\_\_\_\_

Residence:

Citizenship:

Post Office Address:

Full Name of fourth joint inventor:

Signature: \_\_\_\_\_ Date: \_\_\_\_\_

Residence:

Citizenship:

Post Office Address:

002090-99568560